Neural Stack Writeup

December 17, 2017

Contents

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.

[*Dataset description like earlier paper(s)*]

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

[*Something similar to earlier papers*]

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 [*Elaborate more on the CDC loss*].

Other than the probability distributions, the ensemble models also have information about the current week (the week they are merging probabilities for) of the season. Since week goes from 1 to 52/53, to preserve continuity, we encode each week (w) in a vector $\bar{w}$ of two elements in a sinusoidal representation as follows:

$$\bar{w} = [\sin(\frac{2\pi w}{n}), \cos(\frac{2\pi w}{n})]$$

Where n is the total number of weeks (52/53) in the season year. For each neural network model, we create a *with-week* variant which takes in $\bar{w}$ as one of its input.

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. A general description of neural network follows:

1. **Neural Networks**

   Neural networks (or Artificial Neural networks) are machine learning tools modelled loosely after the way neurons are connected in animal brains. The high level aim is to learn a mapping from input to output which may be non-linear. In a neural network, the *neurons* are arranged in some number of *hidden layers* along with an input and an output layer [TODO: Add figure].

In the most general case of a feedforward neural network, the neurons in i^{th} layer have incoming connections from all the neurons in $(i - 1)^{th}$ layer and outgoing connections to all the neurons in $(i + 1)^{th}$ layer. Each neuron in itself collects *its* input values (also called activations of the input neurons), uses its personal set of weights to find a weighted sum of them and passes the result through an activation function to produce its activation value. [TODO: Add figure]. The whole pipeline effectively results in a mapping from input to output parametrized by the neuron connection weights.

To actually fit a model for the input and output, the network needs to train its weights so that it minimizes a certain loss function. The loss function is problem dependent and describes how poorly the output of the network matches with the actual output for the same input. This training is done using backpropagation which is a simple application of chain rule of differentiation for propagating the gradient of loss function to all the neurons so that their weights may be nudged in the right direction for reaching lower loss values. [TODO: Need more explaining on SGD?]

Neural networks have been successful on a variety of tasks [TODO: Cite]. Recent advancements in the techniques and tooling have made it possible to train very *deep* networks capable of learning highly non-linear mappings with high generalization. A short review of these *deep learning* methods is presented in [2].

2. **Mixture density network**

A mixture density network [1] is a simple feed forward neural network which outputs parameters for a mixture of distributions. The loss function here is the crossentropy loss between the mixture of distributions generated by the network and one-hot representation of the truth.

The model we use *assumes* the output from the component models as normally distributed with certain mean and standard deviation. This translates to assuming a single gaussian peak in the output probability distribution from the inputs. It takes in these two inputs (mean and standard deviation of the distribution) from each of the component models and returns a mixture of n gaussians by outputting a set of means (μ_i), standard deviations (σ_i) and weights (w_i) for each distribution in the mixture. The final distribution for a network outputting n mixtures 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. Figure [@fig:mdn] shows the structure of a mixture density model (with weeks) [*REFER TO GITHUB*]

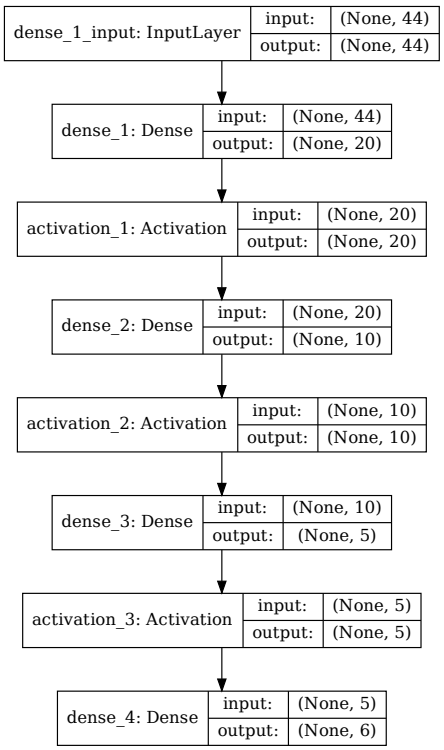

Figure 1: Graph of the mixture density network model. This specific network takes in means and standard deviations of 21 component models (42 inputs) and 2 inputs encoding week. It outputs 6 parameters to be interpreted as weights, means and standard deviations for a mixture of 2 gaussians.

3. **Convolutional neural network**

A convolutional neural network (CNN) [TODO: cite and refer] are neural networks characterized generally by presence of convolutional layers. These layers differ from the regular fully connected layers in that the neurons here are only connected to *local* patches in previous layers. Each convolutional layer has a set of *locally responsive* filters [TODO: image and connected description].

The CNN model in our work puts less assumptions on the input and output distributions and uses a set of 1-dimensional convolutional layers over the complete discrete input distributions. As the output, it outputs the complete discrete probability distribution vector. Figure [@fig:cnn] shows the structure of a convolutional model with weeks [TODO: *REFER TO GITHUB*]

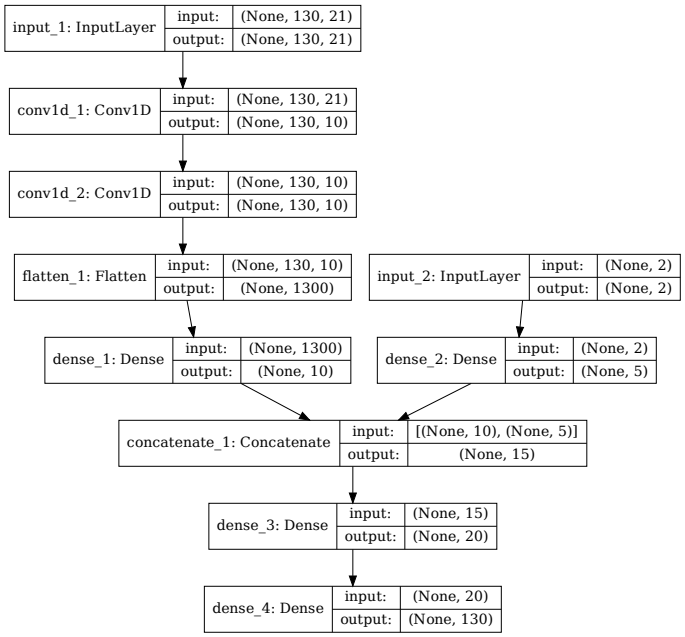

Figure 2: Graph of a convolutional neural model for wili target. The input on the left branch is a set of probability distributions (130 bins) representing wili values for 21 component models. The right branch takes in encoded weeks as vector of size 2. The model finally outputs a probability distribution using 130 bins (same as the component models).

1.3 Results

What did you find?

1.4 Discussion

What does it all mean?

1.5 Conclusions

References

[1] Christopher M Bishop. "Mixture density networks". In: (1994).

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.