Chapter 1

State of the art

In this section, Deep Learning and neural networks are briefly introduced. In addition to that, both the text-to-speech and sentiment analysis tasks are also introduced and discussed, including the challenges that they present as well as the classical and more modern techniques that tackle them, and how they have influenced the development of this project. By the end of this chapter, the core concepts necessary for the discussion of the work done in this publication will have been introduced to the reader.

1.1 Deep Learning

In recent years, deep structured learning, or more commonly called deep learning is a set of techniques that has risen in popuarity and established itself as a new area of machine learning among the scientific and research communities. [4]

[1] discussed the limitations of shallow architectures and provided a theoretical basis to the advantages of deeper ones. But it has only been in recent years that such architectures have been able to be put into production with ever-growing pupularity thanks to compute technologies being every time more available and affordable to consumers.

With deep learning architectures, it is possible to learn complex non-linear representations from large amounts of data. Raw data can be processed and turned into several levels of higher level representations to make it possible to reason about them and make tasks such as classification easier and more effective.

In addition to more compute power, large datasets that can take adavantage of deep models such Imagenet [6] are being made publicly available. These

Hello figure

Figure 1.1: Figure caption goes here!

datasets are often used as benchmarks and serve as points of reference to drive research and development forward.

In summary, one could describe Deep Learning as models capable of obtaining high level information from low level one, and the compute resources and databases that make training of these models possible.

1.2 Deep Neural Networks

In deep learning, a common architecture is the so called deep neural network, which is an artificial neural network with a large number of hidden layers. The desire to decomposing a signal into higher levels of abstraction drives such neural networks configurations, and since more layers require more trainable parameters, large amounts of training data are also needed to train them.

To begin this section, the basic unit of a neural network is introduced to solve a classifiation problem. After that, we see how these basic units can be combined to learn more complex relationships and deal with more generic distributions of data.

1.2.1 From the perceptron to neural networks

The basic unit of a neural network is the neuron. A neuron is defined by a set of scalar inputs (including a bias constant term), a linear operation of these inputs, and an activation function at the output (Figure 1.1). This configuration is also called Perceptron.

The Perceptron is therefore defined by a set of parameters: the weights of the linear operation and the bias term. Equation 1.1 shows the operation that takes place, where $\mathbf{x}$ is the input vector, $\mathbf{w}$ are the input weights, b is the bias term, σ is the diferentiable activation function, often a *sigmoid* (1.2) or a *tanh*, and y is the output. For the sake of simplicity, the remainder of this section assumes σ to be a sigmoid.

$$\mathbf{y} = \sigma(\mathbf{x}^T \cdot \mathbf{w} + b) \tag{1.1}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.2}$$

Hello

Figure 1.2: Model composed of hierarchicaly stacked neurons (neural network)

Assume a binary classification task, where the inputs yielding an activation above a certain defined threshold belong to one class. The perceptron has an intuitive geometrical interpretation. The weights of the equation $w_1, w_2, ...w_n$ and the bias term b represent the rotation translation of the decission boundary that's splits the hyperspace in two regions.

By stacking several of these neurons we can build arbitrary functions create a mapping function from two sapces of arbitrary dimensionalities capable of representing more complex input-output relationships (1.3).

$$\mathbb{R}^N \rightarrow \mathbb{R}^M \tag{1.3}$$

1.2.2 Network Optimization

A neural network is defined by a set of weights and biases as its parameters. These parameteres have to be optimized as to maximize the performance in a classification or regression task. For this we define a function that measures the cost of the task (the error being made) as a function of the inputs and weights and biases of the neural network, and iteratively update each weight w_i according to Equation 1.4, where L is the loss function, n is the iteration step and λ is a scalar parameter called learning rate.

$$w_i[n+1] = w_i[n] - \lambda \frac{\partial L(\mathbf{x}; \mathbf{w}[n])}{\partial w_i} \tag{1.4}$$

The convergence of the loss function is conditioned by the optimization. Nowadays there are techniques that improve the convergence of the loss function such as Adadelta [7] and Adam [5].

The network is optimized using stochastic gradient descent [2], following Equation 1.4. In order to compute the gradients of the loss function, we use the Back-Propagation algorithm [3], which applies the chain rule to compute the gradient of the weights in one layer given the gradients of the pervious layers.

1.3 Recurrent neural networks

Recurrent neural networks are a special type of architecture where the outputs of a hidden layer are fed back to the inputs of the layer. This transforms

Equation 1.1 into Equation 1.5:

$$\mathbf{y}[\mathbf{n}+\mathbf{1}] = \sigma(\mathbf{x}^T \cdot \mathbf{w} + \mathbf{y}[\mathbf{n}]^T \cdot \mathbf{u} + b) \tag{1.5}$$

Where $\mathbf{u}$ is an additional set of trainable weights for each neuron. This introduces a state in each layer and the network can be optimized to model temporal relations, such as predicting the next value in a sequence of numbers given its past context.

This has a problem however that wasn't mentioned in the previous section but is even more apparent in recurrent networks. As the network is unrolled, the back-propagation has to take more steps the longer the temporal dependencies last. This makes the network behave as a deep neural network. In each back-propagation step, the gradient of the loss function is multiplied and if this value is multiplied several times by values smaller than 1, it eventually becomes too small for the network to optimize the earlier layers.

These problems lead to recurrent neural networks to not be trained well enough. To solve this issue, there are special kinds of recurrent neurons that solve this problem.

1.3.1 Long Short-Term Memory (LSTM)

1.3.2 Gated Recurrent Unit (GRU)

Bibliography

[1] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[3] Yves Chauvin and David E Rumelhart. *Backpropagation: theory, architectures, and applications*. Psychology Press, 1995.

[4] Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

[5] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[7] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.