CS5542 Big Data Analytics and Apps

Problem Set-5 (PS-5)

Deadline: May 1 (T), 2017

Submit a hard copy of your solutions to the instructor during the class

Name: Mouqing Jin

Class ID: 8

Part A:

Answer the questions by referring to the following site.

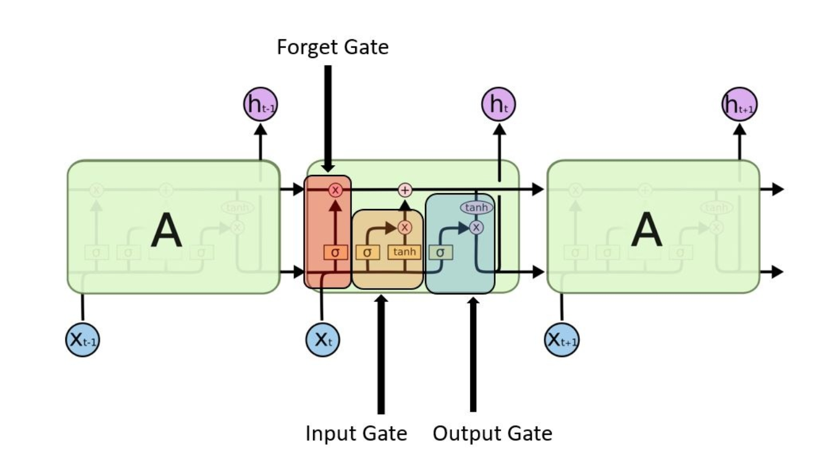
<https://wiki.tum.de/display/lfdv/Recurrent+Neural+Networks+-+Combination+of+RNN+and+CNN>

1. How does LSTM improve Recurrent Neuron Networks?

The LSTM is designed to overcome the error back flow problems through carousels in their special units. This is all done with still a low computational complexity of O(1) and additionaly the LSTM impoves the RNN with the ability to bridge time intervals.

1. How does the LSTM work?

Each LSTM block consists of a forget gate, input gate and an output gate. In figure 5 on the bottom a basic LSTM cell with a step wise explanation of the gates is shown and on the top an other illustration of the cell connected into a network is shown. In the first step the forget gate looks at ht−1 and xt to compute the output ft which is a number between 0 and 1. This is multiplied by the cell state Ct−1 and yield the cell to either forget everything or keep the information. For example a value of 0.5 means that the cell forgets 50% of its information. In the next step the input gate is computing the update for the cell by first multiplying the outputs it and C~t and then adding this output to the input Ct−1∗ft, which was computed in the step before. Finally the output value has to be computed, which is done by multiplying ot with the tanh of the result of the previous step, which yields to: ht=ot∗tanh(Ct) and ot=σ∗(Wo[ht−1,xt]+bo). The formulas are also shown in figure 5 and are displayed in LSTM.



1. What is the complexity of the LSTM algorithm?

The computational complexity is O(1)

1. Discuss how to combine Recurrent and Convolutional Neural Networks.

Recurrent and Convolutional Neural Networks can be combined in different ways. In some paper Recurrent Convolutional Neural Networks are proposed. There is a little confusion abouts these networks and especially the abbreviation RCNN. This abbreviation refers in some papers to Region Based CNN (7), in others to Recursive CNN (3) and in some to Recurrent CNN (6). Furthermore not all described Recurrent CNN have the same architecture. In the following, two approaches are described in more detail. The first approach is described in the paper of Andrej Karpathy and Li Fei-Fei: They connect a CNN and RNN in series and use this for labeling a scene with a whole sentence (14). The second approach from Ming Liang and Xiaolin Hu mixes a CNN with a RNN and use this architecture for better object detection.

1. Compare and contrast the following two approaches

* Approach 1: **CNN and afterwards RNN**

The alignment model described in the paper is a CNN over image region combined with a bidirectional RNN and afterwards a Multimodal RNN architecture, which uses the input of the previous net. This multimodal RNN can finally generate novel descriptions of image regions. In conclusion two single models are combined to one more powerful model, which is used to label images with sentences. To make the architecture more clear the two models are shown in figure 6. The first modal is represented as the left VGGNET and the second module is shown on the bottom right as the RNN.

* Approach 2: **Mixed CNN and RNN**

In a mixed CNN and RNN architecture the positive features of a RNN are used to improve the CNN. Liang and Hu are describing an architecture for object detection in (6) and in (2) a similar architecture for scene labeling is proposed. In these papers the combined network is called RCNN. The following quote describe what their main idea is:

A prominent difference is that CNN is typically a feed-forward architecture while in the visual system recurrent connections are abundant. Inspired by this fact, we propose a recurrent CNN (RCNN) for object recognition by incorporating recurrent connections into each convolutional layer. (6)

The key module of this RCNN are the recurrent convolution layers (RCL), which introduce recurrent connection into a convolution layer. With these connections the network can evolve over time though the input is static and each unit is influenced by its neighboring units. This property integrates the context information of an image, which is important for object detection. The importance of context information is shown in figure 8. In this figure it is very hard to recognize the mouth or the nose without the context (6).

1. Write about the concluding remarks on the contribution of RNN/LSTM/Combination of CNN/RNN

Recurrent Networks are very exciting and have already a very long history. In this history there researchers were able to get a good understanding and feeling about the recurrent network. The fact that it is biological inspired is very promising for getting better performance out of RNN. Furthermore the basic idea of the RNN evolved over the time and many remarkable contributions were made. For example the LSTM, which enhances many properties of the basic RNN. In future it can be assumed that the combination of RNN with other networks, especially the CNN, will be continued. The improvement and the ability to handle sequential data enhance the CNN a lot and brings new unexplored behavior. This is an exciting and promising area of artificial intelligence.

Part B:

1. ***Summary each of the following Reinforcement Learning with Tensorflow series.***
   1. [Part 0 — Q-Learning Agents](Part%200 — Q-Learning%20Agents)**:** <https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-0-q-learning-with-tables-and-neural-networks-d195264329d0>

They are going to be exploring a family of RL algorithms called Q-Learning algorithms. It will hopefully give an intuition into what is really happening in Q-Learning that we can then build on going forward when we eventually combine the policy gradient and Q-learning approaches to build state-of-the-art RL agents

* 1. [Part 1 — Two-Armed Bandit](https://medium.com/@awjuliani/super-simple-reinforcement-learning-tutorial-part-1-fd544fab149)

Reinforcement learning provides the capacity for us not only to teach an artificial agent how to act, but to allow it to learn through it’s own interactions with an environment. By combining the complex representations that deep neural networks can learn with the goal-driven learning of an RL agent, computers have accomplished some amazing feats, like beating humans at over a dozen Atari games, and defeating the Go world champion.

* 1. [Part 1.5 — Contextual Bandits](https://medium.com/@awjuliani/simple-reinforcement-learning-with-tensorflow-part-1-5-contextual-bandits-bff01d1aad9c#.uzs1axw0s)

There is actually a set of problems in-between the stateless situation and the full RL problem. I want to provide an example of such a problem, and show how to solve it. My hope is that those entirely new to RL can benefit from being introduced to each element of the full formulation step by step. Specifically, in this post I want to show how to solve problems in which there are states, but they aren’t determined by the previous states or actions. Additionally, we won’t be considering delayed rewards. All of that comes in Part 2. This simplified way of posing the RL problem is referred to as the Contextual Bandit.

* 1. [Part 2 — Policy-Based Agents](https://medium.com/@awjuliani/super-simple-reinforcement-learning-tutorial-part-2-ded33892c724)

In this post, I am going to describe how we get from that simple agent to one that is capable of taking in an observation of the world, and taking actions which provide the optimal reward not just in the present, but over the long run. With these additions, we will have a full reinforcement agent.

Environments which pose the full problem to an agent are referred to as Markov Decision Processes (MDPs). These environments not only provide rewards and state transitions given actions, but those rewards are also condition on the state of the environment and the action the agent takes within that state. These dynamics are also temporal, and can be delayed over time.

* 1. [Part 3 — Model-Based RL](https://medium.com/@awjuliani/simple-reinforcement-learning-with-tensorflow-part-3-model-based-rl-9a6fe0cce99)

In this case, a model is going to be a neural network that attempts to learn the dynamics of the real environment. For example, in the CartPole we would like a model to be able to predict the next position of the Cart given the previous position and an action. By learning an accurate model, we can train our agent using the model rather than requiring to use the real environment every time. While this may seem less useful when the real environment is itself a simulation, like in our CartPole task, it can have huge advantages when attempting to learn policies for acting in the physical world.

1. Describe the strength and weakness of deep Reinforcement Learning

The methods are discussed here in the context of a Q-Network, but can be applied to Policy Networks as well. To make things more intuitive, I also built an interactive visualization to provide a better sense of how each exploration strategy works (It uses simulated Q-values, so there is no actual neural network running in the browser — though

Weakness:

Failures are necessary, so must train on a very accurate model or a real-world system where failures are not costly. Requires feedback for actionsSufficient possible actions to make a deep learning approach optimal. A short enough lag between action and evaluation to collect enough samples to feed a deep learning model

Strengths :

Does not require a well-structured dataset. In effect does automated dimensionality reduction, so can work on systems with many actions to chose from. Deep reinforcement learning is good at non-linear systems. Can adapt to changing systems - even if they change in complex or unforeseen ways. Can operate on real-world inputs and outputs or can operate from a model (faster but requires an accurate model)