Assignment-14

Dhanush V Nayak
ee23btech11015@iith.ac.in

1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) differ from feedforward networks as they allow cycles in the computation graph. These cycles include delays, enabling the network to retain memory, so earlier inputs influence later outputs. RNNs are primarily used to analyze sequential data, where each time step t introduces a new input vector x_t, and the hidden state z_t captures all relevant information from previous time steps.

1.1 Training a Basic RNN

A basic RNN consists of an input layer, a hidden layer with recurrent connections, and an output layer. The hidden state z_t is updated using:

$$z_t = g_z(W_{z,z}z_{t-1} + W_{x,z}x_t)$$

The predicted output $\hat{y}_t$ is calculated as:

$$\hat{y}_t = g_y(W_{z,y}z_t)$$

Training RNNs involves unrolling the network over time steps to create a deep feedforward network and using backpropagation through time. This allows gradients to be computed and weights updated simultaneously for all time steps. However, the gradients can become unstable, leading to either the vanishing or exploding gradient problem.

For a simple RNN, the gradient with respect to a hidden unit weight $w_{z,z}$ is computed recursively as:

$$\frac{\partial z_t}{\partial w_{z,z}} = g_z'(in_{z,t})\left(z_{t-1} + w_{z,z}\frac{\partial z_{t-1}}{\partial w_{z,z}}\right)$$

This recursion highlights the difficulty of learning long-term dependencies due to the cumulative effects on the gradient.

1.2 Long Short-Term Memory RNNs

To address the limitations of basic RNNs, the Long Short-Term Memory (LSTM) architecture was introduced. LSTMs are designed to retain information over many time steps through memory cells. Unlike standard RNNs, LSTMs allow for constant error flow to prevent vanishing or exploding gradients.

An LSTM includes several gates: the forget gate, input gate, and output gate. The update rules for the LSTM are as follows:

$$
\begin{aligned}
f_t &= \sigma(W_{x,f}x_t + W_{z,f}z_{t-1}) \\
i_t &= \sigma(W_{x,i}x_t + W_{z,i}z_{t-1}) \\
o_t &= \sigma(W_{x,o}x_t + W_{z,o}z_{t-1}) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_{x,c}x_t + W_{z,c}z_{t-1}) \\
z_t &= \tanh(c_t) \circ o_t
\end{aligned}
$$

These gates regulate the flow of information, allowing LSTMs to handle long-term dependencies effectively. LSTMs have been successfully applied to tasks like speech and handwriting recognition.

2 Unsupervised Learning and Transfer Learning

In contrast to supervised learning, which requires labeled data, unsupervised learning focuses on learning from unlabeled data by discovering underlying patterns or representations. Transfer learning allows models to transfer knowledge gained from one task to another. Below are key approaches in unsupervised and transfer learning.

2.1 Autoencoders

Autoencoders are models consisting of two parts: an encoder that maps input data x to a latent representation z, and a decoder that reconstructs the original data from this latent representation. The goal is to minimize the difference between the input and the reconstructed data. A simple form of autoencoders uses linear transformations for encoding and decoding, while more advanced versions employ deep neural networks.

A key extension of autoencoders is the variational autoencoder (VAE), which introduces a probabilistic framework. Instead of learning a deterministic latent representation, VAEs learn a distribution over latent variables. The model is trained by maximizing the evidence lower bound (ELBO):

$$
L(x, Q) = \log P(x) - D_{KL}(Q(z)||P(z|x))
$$

2.2 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) consist of two networks: a generator and a discriminator. The generator creates synthetic data, while the discriminator tries to distinguish between real and generated data. The generator learns to create realistic data by trying to fool the discriminator, while the discriminator improves its ability to differentiate between real and fake data. The two networks are trained simultaneously in a minimax game framework.

GANs have been particularly successful in generating realistic images and videos.

2.3 Deep Autoregressive Models

Deep autoregressive models are used to predict each element of a data sequence based on previous elements. These models do not use latent variables. Instead, they represent the probability of a data sequence as a product of conditional probabilities:

$$
P(x) = \prod_{i=1}^{n} P(x_i|x_1, x_2, \ldots, x_{i-1})
$$

The maximum likelihood solution is given by the Yule-Walker equations, which are closely related to normal equations. An example of a deep autoregressive model is DeepMind's WaveNet, which generates speech by modeling the probability distribution of each audio sample conditioned on previous samples.

2.4 Unsupervised Translation

Unsupervised translation involves transforming data from one domain to another without using paired examples. In unsupervised image translation, for instance, a model learns to map images from one domain (e.g: night scenes) to another (e.g: day scenes) without explicitly pairing night and day images.

GANs are often used for this task, where the generator learns to produce realistic images in the target domain given input from the source domain. This can be extended to natural language translation, where the model learns mappings between languages without parallel corpora.

2.5 Transfer Learning and Multitask Learning

Transfer learning allows an agent to use knowledge gained from one task to improve performance on a different, but related, task. This is particularly useful when labeled data for the new task is limited. For neural networks, transfer learning involves copying weights from a model trained on task A to initialize the model for task B. Pretrained models like ResNet-50 are commonly used for this purpose, where the initial layers are often frozen, and only the higher layers are fine-tuned for the new task.

Multitask learning is a form of transfer learning where a single model is trained on multiple related tasks simultaneously. By sharing knowledge across tasks, the model learns a common representation that is useful for all tasks. This approach often leads to better generalization and performance across different tasks. For example, a natural language processing model could be trained simultaneously on part-of-speech tagging, document classification, and question answering.

3 Applications

Deep learning has found applications in various fields, including computer vision, natural language processing, and reinforcement learning.

3.1 Vision

Deep learning has a huge impact on computer vision, especially with the success of convolutional neural networks (CNNs). The breakthrough came in 2012 when AlexNet won the ImageNet competition, achieving a top-5 error rate of 15.3%, outperforming other methods. AlexNet had five convolutional layers interspersed with max-pooling layers, followed by three fully connected layers. Since then, improvements in CNN architectures have brought the error rate down to less than 2%, surpassing human performance.

CNNs are now used in a variety of vision applications, such as object detection, autonomous driving, and even cucumber sorting.

3.2 Natural Language Processing

In NLP, deep learning enables end-to-end learning, allowing systems to learn directly from data without relying on predefined rules. Machine translation is a prominent example, where a model translates sentences from one language to another based on learned representations. As of 2020, machine translation systems are approaching human performance for language pairs such as French and English.

Word embeddings, which represent words as vectors in a high-dimensional space, have been crucial for NLP models. These embeddings capture the contextual meaning of words, enabling better generalization across tasks such as sentiment analysis and question answering.

3.3 Reinforcement Learning

Reinforcement learning (RL) is another area where deep learning has made significant strides. In RL, an agent learns by interacting with an environment and receiving rewards for good behavior. Deep RL systems, like DeepMind's DQN, have achieved superhuman performance in Atari games.In each case, the agent learned a Qfunction from raw image data with the reward signal being the game score.

Despite these successes, deep RL still has challenges. It can be hard to achieve consistently good results, and trained systems can behave unpredictably if conditions change even slightly. Although it's not widely used in commercial applications yet, deep RL remains a hot topic in research.