RNNs, Unsupervised Learning and Transfer Learning, Applications

Satyanarayana Gajjarapu
AI24BTECH11009
Department of Artificial Intelligence
ai24btech11009@iith.ac.in

1 Recurrent Neural Networks

Recurrent neural networks (RNNs) are distinct from feedforward networks because they allow cycles in the computation graph. RNN has an internal state or **memory**, inputs received at earlier time steps affect the RNN's response to the current input. RNN's update process for hidden state $\mathbf{z}_t$ is given by the equation $\mathbf{z}_t = f_{\mathbf{w}}(\mathbf{z}_{t-1}, \mathbf{x}_t)$, where $f_{\mathbf{w}}$ is a parameterized function and $\mathbf{x}_t$ is a new input vector at each time step. Consider a basic model with an input layer $\mathbf{x}$, a hidden layer $\mathbf{z}$ with recurrent connections and an output layer $\mathbf{y}$. The equations defining the model at time step t:

$$\mathbf{z}_t = f_{\mathbf{w}}(\mathbf{z}_{t-1}, \mathbf{x}_t) = \mathbf{g}_z(\mathbf{W}_{z,z}\mathbf{z}_{t-1} + \mathbf{W}_{x,z}\mathbf{x}_t) \equiv \mathbf{g}_z(\mathbf{in}_{z,t})$$
$$\hat{\mathbf{y}}_t = \mathbf{g}_y(\mathbf{W}_{z,y}\mathbf{z}_t) \equiv \mathbf{g}_y(\mathbf{in}_{y,t})$$

where $\mathbf{g}_z$ and $\mathbf{g}_y$ denote the activation functions for the hidden and output layers respectively. Consider a model which is unrolled for T steps with weight matrices $\mathbf{W}_{x,z}$, $\mathbf{W}_{z,z}$ and $\mathbf{W}_{z,y}$ shared across all time steps. Computing the gradient of the squared-error loss L with respect to the hidden-layer weight $w_{z,z}$ using the chain rule is as follows:

$$\frac{\partial L}{\partial w_{z,z}} = \sum_{t=1}^{T} -2(y_t - \hat{y}_t) g'_y(in_{y,t}) w_{z,y} \frac{\partial z_t}{\partial w_{z,z}}$$
$$\frac{\partial z_t}{\partial w_{z,z}} = g'_z(in_{z,t})(z_{t-1} + w_{z,z}\frac{\partial z_{t-1}}{\partial w_{z,z}}$$

The above equation is recursive in nature. If $w_{z,z} < 1$, RNN will suffer vanishing gradient problem. If $w_{z,z} > 1$, RNN will suffer exploding gradient problem. **Long short-term memory** (LSTM) is a type of RNN. It contains 3 **gating units**
• **forget gate f** determines if each element of the memory cell is remembered or forgotten.
• **input gate i** determines if each element of the memory cell is updated at current time step.
• **output gate o** determines if each element of the memory cell is transferred to the short-term memory $\mathbf{z}$.

2 Unsupervised Learning and Transfer Learning

Unsupervised learning algorithms take a training set of unlabeled examples $\mathbf{x}$. The algorithm should try to learn new representations and generative model. Suppose a joint model $P_W(\mathbf{x}, \mathbf{z})$ is learnt, it achieves both representation learning and generativ modeling when integrated $\mathbf{z}$ out of $P_W(\mathbf{x}, \mathbf{z})$ to obtain $P_W(\mathbf{x})$.

Preprint. Under review.

In **probabilistic principal components analysis** (PPCA) model, $\mathbf{z}$ is chosen from a zero-mean, then $\mathbf{x}$ is generated from $\mathbf{z}$ by applying a weight matrix $\mathbf{W}$ and σ^2 is noise parameter.

$$P(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$$
$$P_W(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{W}\mathbf{z}.\sigma^2\mathbf{I})$$
$$P_W(\mathbf{x}) = \int P_W(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I})$$

Many unsupervised deep learning algorithms are based on the idea of an **autoencoder**. It contains an encoder that maps from $\mathbf{x}$ to a representation $\hat{\mathbf{z}}$ and a decoder that maps from a representation $\hat{\mathbf{z}}$ to observed data $\mathbf{x}$. The model is trained so that $\mathbf{x} \approx g(f(\mathbf{x}))$, where f and g are parameterized functions of encoder and decoder respectively. A very simple autoencoder is the linear autoencoder, where both f and g are linear with a shared weight matrix $\mathbf{W}$:

$$\hat{\mathbf{z}} = f(\mathbf{x}) = \mathbf{W}\mathbf{x}$$
$$\mathbf{x} = g(\hat{\mathbf{z}}) = \mathbf{W}^\top\hat{\mathbf{z}}$$

Other types of unsupervised learning algorithms are **deep autoregressive models**, **generative adversarial networks** and **unsupervised translation**. In **transfer learning**, experience with one learning task helps an agent learn better on another task. Learning rate depend on how similar the both tasks are. Multitask learning is a form of transfer learning in which we simultaneously train a model on multiple objectives.

3 Applications

Deep learning has had a significant impact on **computer vision**, especially with the success of AlexNet in the 2012 ImageNet competition. It used ReLU activation functions and took advantage of GPUs to speed up the process of training 60 million weights. It also has an impact on **natural language processing**(NLP) in the fields of machine translation, speech recognition and word embeddings. It combined with reinforcement learning to form **deep reinforcement learning**, which is used in DeepMind's AlphaGo. Deep RL still faces significant obstacles, it is often difficult to get good performance.