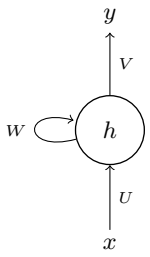
Ludwig-Maximilians-Universitaet Muenchen 28.05.2021
Institute for Informatics
Prof. Dr. Volker Tresp
Rajat Koner
Andreas Lohrer
Sebastian Schmoll

Machine Learning
Summer 2021
Exercise Sheet 6

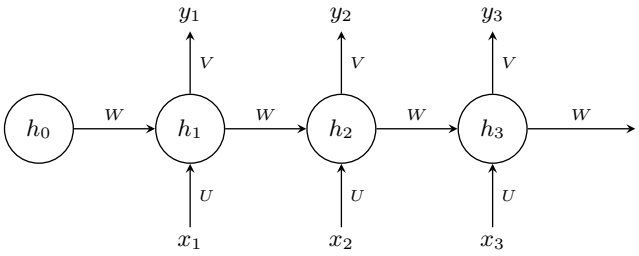
Exercise 6-1 RNNs

Consider the following RNN in which each hidden state h_t is given by $h_t = \sigma(Wh_{t-1} + Ux_t)$ where $\sigma(z) = \frac{1}{1+e^{-z}}$, $h_t, h_{t-1}, x_t \in \mathbb{R}^d$, and $W, U \in \mathbb{R}^{d \times d}$. Assume that $h_0 = 0$ is given as initial hidden state.



(a) Draw the unfolded RNN for three timesteps $t = 1..3$. Do not forget to label the nodes and edges.

Possible Solution



(b) What is the main advantage of RNNs compared to simple feed forward neural networks? What are they commonly used for?

Possible Solution

- RNNs provide a memory for previous inputs, i.e. past inputs are encoded in the hidden states and passed along through time such that they can be used for prediction of the current output (instead of only using the current input).

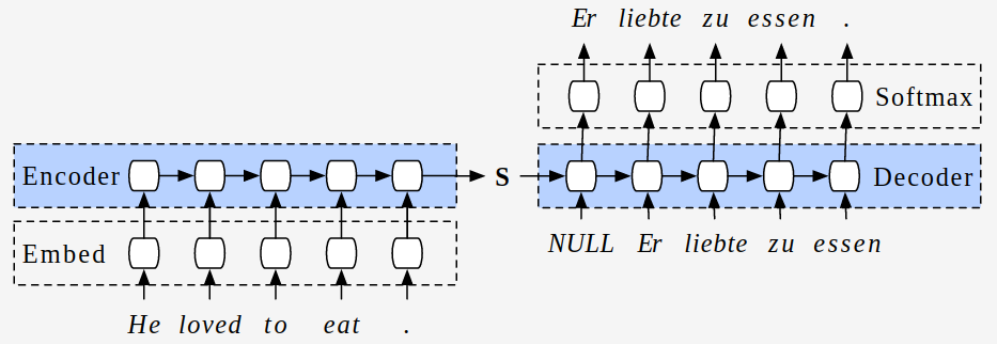
- Commonly used for sequence modelling, e.g. speech recognition, translation, time series prediction etc.

(c) Explain the idea of Encoder-Decoder Networks!

Possible Solution

Encoder: RNN (usually LSTM) with word embeddings as inputs and no output layer. The last hidden state (end-of-sentence) is the encoder vectors

Decoder:
RNN (usually LSTM) with initial latent state = encoder vector
Output: $y_t = \sigma(Vh_t)$ (one-hot encoded word)
Latent state has previous state and embedding of previous output as input: $h_t = \sigma(Wh_{t-1} + Va_{y_{t-1}})$ where a_w is the embedding of a word w.



Exercise 6-2 Attention

The Encoder-Decoder model has found to be limited when it comes to very long sequences:

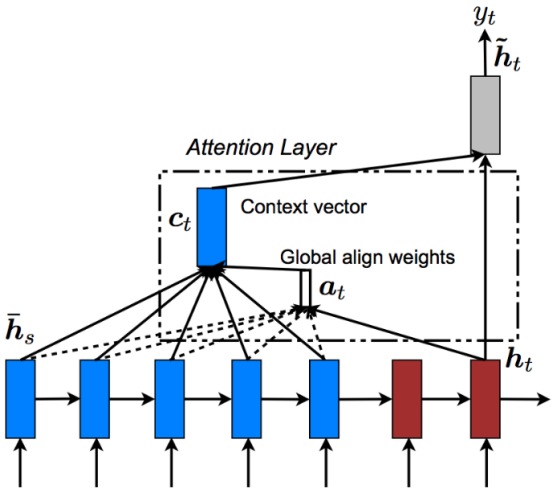
"A potential issue with this encoder-decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus."(Neural Machine Translation by Jointly Learning to Align and Translate, 2015.)

(a) How does the attention mechanism overcome this problem?

Possible Solution

- It provides a richer encoding of the source sequence from which to construct a context vector
- For each output word of the Decoder, the model can learn what encoded words in the source sequence what hidden states of the encoder) to pay attention to and to what degree (instead of just using the last hidden state of the encoder)

(b) Consider an Encoder-Decoder model with global attention as shown in the figure below. The blue hidden units correspond to the encoder and the red ones to the Decoder.



The depiction of Global Attention in an Encoder-Decoder Recurrent Neural Network.
Taken from "Effective Approaches to Attention-based Neural Machine Translation."

Assume we want to translate the sentence $x =$ "This is German" into "Dies ist deutsch".
Let the hidden units corresponding to the three input words in the Encoder be given as:
$\bar{h}_1 = \begin{pmatrix}0\\1\\1\end{pmatrix}$, $\bar{h}_2 = \begin{pmatrix}2\\1\\0\end{pmatrix}$ and $\bar{h}_3 = \begin{pmatrix}0\\3\\0\end{pmatrix}$. Moreover, let the current target state of the Decoder corresponding to the output word "deutsch" be $h_t = \begin{pmatrix}0\\2\\0\end{pmatrix}$.

- Calculate the attention scores (alignments) $a_t(\bar{h}_i)$ for $i = 1..3$ using the formula introduced in the lecture. How can the result be interpreted?

Possible Solution

$a_t(\bar{h}_i) = \frac{e^{\bar{h}_i^T h_t}}{\sum_k e^{\bar{h}_k^T h_t}} = \mathsf{softmax}(\bar{h}_i^T h_t)$

We have: $\bar{h}_1^T h_t = 2; \quad \bar{h}_2^T h_t = 2; \quad \bar{h}_3^T h_t = 6.$

Thus: $a_t(\bar{h}_1) = a_t(\bar{h}_2) = \frac{e^2}{e^2+e^2+e^6} = \frac{7.39}{7.39+7.39+418.2} = 0.0177$ and $a_t(\bar{h}_3) = 0.964$

Interpretation: When predicting the word "deutsch" the most important input word is "german" (obviously).

- Calculate the resulting context vector c_t as weighted sum of the hidden units in the Encoder.

Possible Solution

$c_t = \sum_i a_t(\bar{h}_i) \cdot \bar{h}_i = 0.0177 \begin{pmatrix}0\\1\\1\end{pmatrix} + 0.0177 \begin{pmatrix}2\\1\\0\end{pmatrix} + 0.964 \begin{pmatrix}0\\3\\0\end{pmatrix} = \begin{pmatrix}0.035\\2.929\\0.0177\end{pmatrix}$

This context vector is used as input for the attentional hidden state: $\tilde{h}_t = \sigma(Vh_t + Dc_t)$.

Exercise 6-3 PyTorch - Early Stopping and Save/Load Models

On the course website you find an ipython notebook leading you through the former implementation of a CNN in PyTorch. We will extend this model by early stopping and learn how we can save/store the model's parameters.

Exercise 6-4 Autoencoders

(a) What is the main idea of an autoencoder? What components does it consist of and how is it trained?

Possible Solution

Autoencoders aim at learning an inverse feature map $h = g_e(x; \theta_e)$, i.e. an encoder. Such a function cannot be learned directly, since we cannot measure h. The idea of an autoencoder is to learn together with the encoder a decoder function $x = g_d(h; \theta_d)$, such that an input x can be reconstructed as $\hat{x} = g_d(g_e(x))$. The autoencoder can then be trained using a reconstruction loss

$$L(\theta) = \sum_i ||x_i - \hat{x_i}||^2$$

and a gradient-based learning rule. Since the autoencoder is forced to accurately reconstruct the inputs from the learned embeddings, it needs to learn compact and expressive representations of the inputs. Distances in the latent space are thus often more meaningful than in the original feature space.

(b) Name at least three possible applications of an autoencoder.

Possible Solution

- Dimensionality reduction: The dimensionality of the latent space is usually set smaller than the dimensionality of the data space, such that the model learns an informative lower-dimensional representation of the data.
- Pretraining: An autoencoder can be used to pretrain another neural network, e.g. a classifier. For instance, we could take the trained encoder, add a few dense layers for prediction and re-train the resulting model end-to-end with a prediction loss such as the binary cross entropy.
- Outlier detection: The reconstruction error is often large for inputs which are very different from the usual patterns observed in the training data. This can be used for outlier/anomaly detection by simply using the reconstruction error as an outlier score.
- Generative model: The learned decoder can be used as a generative model. Similarly as a GAN, it can be used for generating additional synthetic training samples, which is particularly useful in a setting where only few labeled samples are available. The Variational Autoencoder (VAE) is especially suited for this, since the latent dimensions are independently Gaussian distributed with unit variance, so we can just sample from these distributions and decode the resulting latent vector.

(c) Consider a linear autoencoder (without regularization or denoising). What happens if the number of dimensions of the latent space is set larger than the dimensionality of the data space?

Possible Solution

In this case, the autoencoder will simply learn the identity function and achieve zero reconstruction error. Since the latent space has a sufficient number of dimensions, the autoencoder can just copy the input into the latent space and back. This is a general problem if the autoencoder is given too much capacity. By introducing a bottleneck, the autoencoder is forced to learn compact and expressive representations. Modifications such as Denoising (DAE) or using additional regularization (Sparse AE or CAE) further help avoiding such effects.

Exercise 6-5 Generative Adversarial Networks (GANs)

(a) Provide a short and intuitive, non-formal description of a GAN. What is the main idea? What are the individual components of the model and how do they interact?

Possible Solution

The main idea is to train a generator that is able to produce samples which are similar to the training data observed by the model. In particular, the generator is considered to perform well, if a discriminator is not able to tell the generated samples apart from the training data, i.e. if the generator can successfully fool the discriminator. The GAN consists of the generator and discriminator, which are usually multi-layer perceptrons and are trained jointly.

(b) What are possible applications for a GAN?

Possible Solution

Possbile applications include:

- Similarly as with encoders, the convolutional layers of a trained discriminator can be used to initialize a discriminator for another task, such as image classification, by exchanging the last layer with a linear layer for prediction. This is particularly useful, if only relatively few labeled samples are available for this task.
- The generator can be used to produce high-quality artificial data samples. This is also useful in a setting with few labeled samples, since the generator can be used to generate more synthetic training samples.

(c) Name at least two different variations of the GAN model and provide a possible application for each.

Possible Solution

- Conditional GAN (cGAN): Use the class label as an additional input to the generator and discriminator. This makes it possible to produce samples conditioned on a certain label combination. For instance, given labels extracted from a textual description of a bird such as "white with some black on its head and wings and a long orange beak", the generator can produce plausible images matching that description (reverse captioning).

- InfoGAN: Similar to the cGAN, but the discriminator additionally predicts the class label. A possible application is to not only synthesize novel images, but to modify existing ones, e.g. adding a beard or sunglasses to a face. The InfoGAN improves this task by disentangling different concepts in the latent space. This is achieved by letting the discriminator predict the class label of the samples produced by the generator.
- CycleGAN: Used for unpaired image-to-image translation, e.g. turn horses into zebras without having aligned pairs of horse and zebra images. Train two generators, for mapping horses to zebras and zebras to horses respectively, such that if a zebra is turned into a horse and back into a zebra (analogously for a horse), the original and the resulting zebra are similar. This is called cycle consistency. Additionally, two discriminators are trained which separate real from fake horses and real from fake zebras, respectively. A CycleGAN can be used for many different style transfer tasks, e.g. summer to winter, photo to style of a particular artist and so on.