6.874/Equivalent Recitation 2

Tess Gustafson

February 27, 2021

Topics For Today

- Transformers
- Types of Neural Networks and When to Use Them
- Convolutional Neural Networks and Feature Extraction
- Recurrent Neural Networks Overview Resources

1 Transformers

Transformers are a new type of deep learning model (specifically introduced in 2017), that are used primarily in the field of natural language processing (NLP). A good compare and contrast of Transformers versues regular RNNs would be:

- Transformers are designed to handle sequential data for tasks such as translation and text summarization (like RNNs with natural language)
- Transformers do not require that the sequential data be processed in order
  - If the input data is a natural language sentence, the Transformer does not need to process the beginning of it before the end
  - This means that the Transformer allows for much more parallelization than RNNs
  - This reduces the training times of the model

How Transformers Work

- *Attention*
  - The attention-mechanism looks at an input sequence and decides at each step which other parts of the sequence are important
  - Ex: When reading this text, you always focus on the word you read but at the same time your mind still holds the important keywords of the text in memory in order to provide context
- Encoder and decoder like LSTMs, but attention plays into one of the critical differences between Transformers and RNNs

  - If we have an encoder and decoder like an LSTM that is trying to translate a sentence from one language to another, imagine that instead of only writing down the translation of the sentence in the imaginary language, the Encoder also writes down keywords that are important to the semantics of the sentence
  - Those new keywords are given to the Decoder and make the translation much easier for the Decoder because it knows what parts of the sentence are important and which key terms give the sentence context.
- Architecture
  - No recurrent networks like LSTM
  - Encoder and Decoder are composed of modules that can be stacked on top of each other multiple times, which is described by Nx in the figure shown in lecture.
  - Inputs and outputs (target sentences) are first embedded into an n-dimensional space since we cannot use strings directly.

Resources

For further explanations please visit `https://arxiv.org/pdf/1706.03762.pdf` for the published research paper regarding Transformers.

Also read about BERT, a Bidirectional Transformer that has vastly improved Natural Language Tasks here: `https://arxiv.org/pdf/1810.04805.pdf`.

2 Types of Networks and When to Use Them

2.1 Artificial Neural Networks (ANNs)

An artificial network consists of multiple perceptrons/neurons at each layer and can be thought of as a Feed-Forward neural network because the inputs are processed only in the forward direction.

2.1.1 Types of Problems ANNs Can Solve

- Problems pertaining to tabular data
- Problems pertaining to image data
- Problems pertaining to text data

2.1.2 Advantages of ANNs

- ANNs are capable of learning any nonlinear function
  - ANNs have the capacity to learn weights that map any input to the output
  - Because of this, ANNs are often referred to as **Universal Function Approximators**
- ANNs, like other deep networks, can use many different activation functions
  - These introduce nonlinear properties to the network

  - These help the networs learn any complex relationships between inputs and outputs
  - Without activation functions, the network can only learn linear relationships and never complex relationships

2.1.3 Disadvantages of ANNs

- ANNs can struggle with image classification problems since you have to convert 2- (or 3-) dimensional images into 1-dimensional vectors before training the model
  - This causes ANNs to lose spatial features of an image
  - This also increases the number of trainable parameters drastically
- ANNs cannot capture sequential information in the input data which is required for dealing with sequence data

As you will see coming up, these disadvantages can be overcome with other network types!

2.2 Convolutional Neural Networks (CNNs)

A convolutional neural network is a powerful neural network that uses filters to extract features from images in a way that position information of pixels is retained throughout the hidden layers. It does so by using kernels to extract the relevant features from the input using convolution operations.

2.2.1 Types of Problems CNNs Can Solve

- Problems pertaining to image data such as object detection, facial recognition, and image segmentation as well as video analysis.
- Also can be used for sequential inputs as well

2.2.2 Advantages of CNNs

- CNNs are able to learn relevant features from an image and can detect relevant features in different locations within a image. This is **feature learning** and **spatial recognition** and conventional neural networks cannot do this.
- Another pro of CNNs in the **parameter sharing**, where weight parameters are shared across different time steps, which can greatly cut down on the number of weights needed to train a model.

2.2.3 Disadvantages of CNNs

- Hyper-parameter tuning is non-trivial and a lot of data is required for training for accurate results.
- Another con is the fact that these networks can become quite large and still have a high computational cost.
- CNNs can have problems such as exploding and gradient vanishing
  - **Exploding gradient** refers to the problem when we have a network of n hidden layers and therefore n derivatives that will be multiplied together. If the derivatives are large then the gradient will increase exponentially as we propagate down the model until they "explode".

  - **Vanishing gradient** refers to the opposite problem. If these derivatives are small, then the gradient will decrease exponentially as we propagate down the model until they "vanish".

2.3 Recurrent Neural Networks (RNNs)

Recurrent neural networks has a recurrent condition on the hidden state that ensures that sequential information is captured in the input data.

2.3.1 Types of Problems RNNs Can Solve

- Problems pertaining to time series data
- Problems pertaining to text data
- Problems pertaining to audio data

2.3.2 Advantages of RNNs

- RNNs capture the sequential information present in the input data
  - The output at each time step depends not only on the current word (or input) but also on the previous words (or inputs)
- RNNs also use parameter sharing, which results in fewer parameters to train and decreases computational costs
  - This means for larger input sizes, the model size does not increase

2.3.3 Disadvantages of RNNs

- Due to its recurrent nature, the computations can be slow
- Training RNN models can be quite difficult, especially if we use ReLU or tanh as activation functions because these will struggle with longer sequences
- RNNs are prone to problems such as exploding and gradient vanishing

3 Convolutional Neural Networks and Feature Extraction

Convolutional neural networks (CNNs) are neural networks that have at least one convolutional layer and are used mainly for image processing, classification, and segmentation.

3.1 Convolutional Layer Definitions

3.1.1 Depth

Depth corresponds to the number of filters we would like to use. A set of neurons that are all looking at the same region refers to a **depth column**.

3.1.2 Stride

Stride refers to how we slide the filter. When the stride is 1 we move the filters one pixel at a time. When the stride is 2, we move over two pixels at a time. Strides 3 and larger are usually uncommon.

3.1.3 Zero-Padding

Zero-padding refers to padding the input volume with zeros around the border and is useful to control the spatial size of the output volumes. Often times this is used to exactly preserve the spacial size of the input volume so that the input width and height is the same for the output.

3.1.4 Spatial Extent

When dealing wiht high-dimensional inputs like images, it is impractical to connect neurons to all neurons in the previous volume. Instead we will connect each neuron to only a local region of the input volume, where the spatial extent of this connectivity is a hyperparameter called the **receptive field** of a neuron (also known as the filter size).

3.1.5 Parameter Sharing

In CNNs, we use parameter sharing by making one assumption: if one feature is useful to compute at some spatial position (x, y) then it should also be useful to compute at a different position (x_2, y_2). So if we have a single 2-dimensional slice of depth as a depth slice (e.g. a volume of [128x128x3] has 3 depth slices of size [128x128]), then we are going to constrain neurons in each depth slice to use the same weights and bias.

Notice if all neurons in a single depth slice use the same weight vector, then the forward pass of the convolution layer in each depth slice is computed as a convolution of the neuron's weights with the input volume. This is why we call these filters or kernels.

3.2 Convolutional Layer Equations

- Input volumes of size $W_1 \times H_1 \times D_1$
- Needs four hyperparameters
  - The number of filters K
  - The spatial extent of the filters F
  - The stride S
  - The amount of zero padding if any P
- The number of weights per filter is $F \cdot F \cdot D_1$ with parameter sharing, giving us a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases
- The output volume is size $W_2 \times H_2 \times D_2$
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$
  - $D_2 = K$

- The d=th depth slice (of slice $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input with a stride of S and then offset by the d-th bias

3.3 Pooling Layers

Pooling layers are layers often put in-between successive convolutional layers to reduce spatial size of the representation and to reduce the number of parameters in the network. This also helps control overfitting. Generally pooling layers use the max function to take the largest value of the filter as we slide the filter along our input volume. Some other pooling layers use average pooling or L2-norm pooling but max pooling works better in practice.

3.3.1 Pooling Equations

- Input volumes of size $W_1 \times H_1 \times D_1$
- Needs two hyperparameters
  - The spatial extent of the filters F
  - The stride S
- The output volume is size $W_2 \times H_2 \times D_2$
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = K$
- It is not common to add zero-padding for max pool layers
- No parameters are introduced since it computes a fixed function of the input

3.4 CNN Architecture

The most common CNN architecture uses a few stacks of convolutional layers with ReLU activation followed by a pooling layer and then repeating this until the image has been merged spatially to a small size. Often times after this size has been shrunk, people will add fully connected layers, with the final fully connected layers being the outputs of classification scores. Activations for the final output are often softmax since all neuron outputs for imaging are usually predicting probabilities that need to sum up to one.

3.5 Resources

For further explanations about convolutional neural networks please visit `https://cs231n.github.io/convolutional-networks/` for a very detailed but simplified explanation.

4 Recurrent Neural Networks Overview Resources

Unfortunately there is not enough time to go over RNNs in more detail this recitation, but here are some good resources for learning more about RNNs:

https://towardsdatascience.com/
illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

https:
//stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

http://www.wildml.com/2015/09/
recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

https://towardsdatascience.com/
illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21