CSCI 5922 - Problem Set 3

Karan Praharaj

January 19th, 2022

1. Regularization

The process of modifying a machine learning model to favor "simpler" prediction rules over more complex rules, so as to reduce the possibility of overfitting, is called regularization. Conventionally, this entails modifying the original loss function to apply penalties to certain values of the parameters that are being learned. The weights that are large are usually penalized most. In summary, complicated hypotheses lead to overfitting, and so to penalize hypothesis complexity, we modify the error function and control this penalty mechanism with a regularization coefficient.

2. CNN features

Low-level features encode, as the term suggests, the lowest level details of the image, which are learned in the initial stages of the network. These features carry basic structural information like lines, curves, edges, simple shapes and dots. Mid level features are found at the intermediate stage of convolutional networks and they are composed of the preceding low-level features. These mid-level features combine the low-level information and detect more complex edges and shapes. Finally, the high-level features in the final stages of the CNN are built on top of the mid-level features to detect yet more abstract concepts and larger shapes in the image.

3. Fine-tuning

Two advantages of fine-tuning a neural network over training a network from scratch are:

- Fine-tuning a pre-trained network can be done with a relatively small amount of data than would be needed for training a network from scratch. Thus, fine-tuning negates the need to collect a large dataset.
- Because fine-tuning is done over a smaller amount of data, it is faster than training a network from scratch.
- Fine-tuning reduces the need for heavy computational resources that are often crucial for training a network from scratch in a reasonable duration.

4. Computer Vision Problems

Image Classification : The task of classifying an image as a whole and assigning it a particular label is called image classification. In this problem, separate regions or sub-parts of the image are not assigned different labels. Rather, the image in its entirety is assigned one single label.

Object detection : This task is concerned with identifying all instances of one or more objects that may be present in an image. Conventionally, it involves generating a rectangular bounding box around the parts of the image where an object is detected. Because the bounding boxes are always rectangular, irrespective of the actual shape of the object, we cannot reliably compute the area or perimeter of the object in the image using this technique.

Semantic Segmentation : Semantic segmentation is a computer vision task which involves pixel-wise comprehension of the image. Each pixel in the image is assigned a label from a pre-defined list of category classes, and contiguous sets of pixels assigned to the same label are said to be an instance of the corresponding object. Semantic segmentation usually also has a 'background' class for pixels that are not a constituent of any object.

In image classification, the final layer classifies the image as a whole, and all the features are treated as features from one entity. On the other hand, in object detection, multiple classifications are generated for each region of interest that is proposed. Finally, for semantic segmentation, the last layer performs a pixel-wise prediction by generating a probability for each class from the list of possible classes, assigning to that pixel the label corresponding to the class with maximum probability.

5. Recurrent Neural Networks

(a) Two advantages of using recurrent layers instead of fully connected layers are as follows:

- Recurrent neural networks are the only flavor of neural networks which have the ability to encode temporal information. This ability enables them to model sequential data or time-series data. While fully connected neural networks assume that inputs and outputs are independent of each other, recurrent networks can model scenarios where the output at one stage depends on the states and outputs in previous stages.
- The other advantage RNNs have over fully connected neural networks is that they share the same parameters (V, W and U) in each layer throughout the entire network. This helps in keeping the model size low, because the model size does not increase as the inputs get longer.
- They can, in theory, handle any input size by just recurrent "unfolding" to have more/less time steps depending on the input. This is not the case for fully-connected networks.

(b) Recurrent layers differ from convolutional layers in the following ways:

- The fundamental difference between convolutional neural networks and recurrent neural networks is that convolutional networks have the ability to process spatial information, whereas recurrent networks are designed to process sequential or temporal information.
- Recurrent neural networks can handle an arbitrary length of input and output, whereas convolutional neural networks accept a fixed input size and return a fixed output size.

Recurrent neural networks should be opted for over convolutional neural networks when the data in question has a sequential or temporal character rather than a spatial character. They are suitable to interpret data where the output at any given stage depends on not only the input at that stage, but also all the past outputs generated until that point in time. If your application is based on sequential or time-series data like temperature forecast, stock price prediction, language translation, part-of-speech tagging or language modeling, then RNNs become the better choice.

(c) *One-to-many* RNNs are designed to handle sequential tasks where the input data has one time-step, and the output has multiple time-steps. In short, it accepts a single input and a sequence of multiple outputs. A good example for its application is image captioning, where the input is one image, and the output is a sequence of multiple words describing the image.

Many-to-one RNNs are designed to handle sequential tasks where the input data is of multiple time-steps, and needs a prediction of a single output. An example for an application of many-to-one RNNs is sentiment analysis or text classification tasks, where a stream of words is accepted as input by the RNN and a sentiment (e.g. - good or bad) or a single class label is generated as output at inference time.

Many-to-many RNNs are designed to handle sequential tasks where the input data and output data both extend over multiple time-steps. An example for its usage is machine translation where the input sentence is of a given number of words in one language, and the output sequence is in the target language of interest, potentially with a different number of words as compared to the input.

(d) Motivation for gated RNNs: The training of vanilla recurrent neural networks has been shown to be challenging for tasks which need the learning of long-term temporal dependencies. This is explained by the phenomenon of vanishing gradients, which causes an exponential decay of the loss function across layers, thus making the model unable to predict long-distance dependencies. To mitigate this, gated networks were introduced with special units called memory cells that are aimed at preserving information learned in previous time steps for use in future time steps, potentially even after a long period of time. This mechanism is facilitated by a set of gates which regulate the flow of information with respect to the network, i.e. when it is saved in the memory, pushed as output, or forgotten/erased. This helps the architecture in learning longer-term dependencies.

Two types of gated RNNs are : 1) LSTMs (Long-short term memory) 2) GRUs (Gated recurrent units)

(e) If the number of input characters to a character sequence predictor RNN increases, it means that the number of time steps increase. But RNNs are designed to use the same model parameters across all time steps, and therefore, the number of model parameters associated with an RNN will not increase as the number of time steps increases. Thus, doubling the number of input characters will lead to no change in the parameterization cost of our 2-layer recurrent neural network.