Predicting radio signal modulation scheme with an ensemble of convolutional neural networks

Benjamin Posnick

1 Introduction

This paper outlines a deep learning approach using ensemble methods to predict the modulation scheme used for encoding a given signal using samples of its in-phase and quadrature (I/Q) components.

2 Data Preprocessing

2.1 Dataset

The training set $\mathbf{X}_{trn} \in \mathbb{R}^{30000\times1024\times2}$ contains 30,000 examples, where each example is a matrix $\mathbf{X}_i \in \mathbb{R}^{1024\times2}$ composed of two vectors $\mathbf{X}_{i,I} \in \mathbb{R}^{1024}$ and $\mathbf{X}_{i,Q} \in \mathbb{R}^{1024}$, representing the in-phase and quadrature components for a given example:

$$\mathbf{X}_i = \begin{bmatrix} \mathbf{X}_{i,I} & \mathbf{X}_{i,Q} \end{bmatrix} \tag{1}$$

The testing set $\mathbf{X}_{tst} \in \mathbb{R}^{20000\times1024\times2}$ is formatted the same, but only contains 20,000 examples.

It is known that deep learning models often require large datasets to obtain good performance. Previous work on this topic by Li *et al* [1] employed samples of size 128. Motivated by this, in order to better utilize the given dataset, each signal in $\mathbf{X}_{trn}$ and $\mathbf{X}_{tst}$ was partitioned into 4 equal-sized sub-samples such that $\hat{\mathbf{X}}_{trn} \in \mathbb{R}^{120{,}000\times256\times2}$ and $\hat{\mathbf{X}}_{tst} \in \mathbb{R}^{80{,}000\times256\times2}$. Each of the 4 partitions were labeled with the same label as the original full-length signal. This was based on the assumption that a given signal could be treated as 4 separate signals that happen to comprise a contiguous sequence. The examples in $\hat{\mathbf{X}}_{trn}$ were treated independently thereafter. However, for each example in $\hat{\mathbf{X}}_{tst}$, each of the 4 partitions were classified separately and the best prediction (based on averaging softmax probabilities) was used to classify the original full-length signal from $\mathbf{X}_{trn}$ (see "Prediction" section).

2.2 Feature Extraction

Using domain knowledge of signals, features were extracted so as to improve the quality of data inputted into the deep learning model. The same process was applied to both $\hat{\mathbf{X}}_{trn}$ and $\hat{\mathbf{X}}_{tst}$ – for brevity, we will denote them both as $\hat{\mathbf{X}}$. Feature matrices $\mathbf{F}_i \in \mathbb{R}^{120{,}000\times256\times4}$ were created for each example $\hat{\mathbf{X}}_i \in \hat{\mathbf{X}}$:

$$\mathbf{F}_i = \begin{bmatrix} \mathbf{F}_{i,I} & \mathbf{F}_{i,Q} & \mathbf{F}_{i,\phi} & \mathbf{F}_{i,A} \end{bmatrix} \tag{2}$$

where $\mathbf{F}_{i,I}$ and $\mathbf{F}_{i,Q}$ are the vectors $\hat{\mathbf{X}}_{i,I}$ and $\hat{\mathbf{X}}_{i,Q}$, respectively, and:

$$\mathbf{F}_{i,\phi} = \arctan2(\mathbf{F}_{i,Q}, \mathbf{F}_{i,I}) \tag{3}$$

$$\mathbf{F}_{i,A} = \sqrt{\mathbf{F}_{i,Q}^2 + \mathbf{F}_{i,I}^2} \tag{4}$$

[1] Li, M.; Li, O.; Liu, G.; Zhang, C. An Automatic Modulation Recognition Method with Low Parameter Estimation Dependence Based on Spatial Transformer Networks. Appl. Sci. 2019, 9, 1010.

CLDNN Architecture			
#	**Layer**	**Parameters**	**Activation Function**
1	1D-Convolution	64 filters, 3×1 kernel	SeLU, Batch Normalization
2	1D-Max-Pooling	2×1 pool	None
3	1D-Convolution	64 filters, 3×1 kernel	SeLU, Batch Normalization
4	1D-Max-Pooling	2×1 pool	None
5	1D-Convolution	64 filters, 3×1 kernel	SeLU, Batch Normalization
6	1D-Max-Pooling	2×1 pool	None
7	1D-Convolution	64 filters, 3×1 kernel	SeLU, Batch Normalization
8	1D-Max-Pooling	2×1 pool	None
9	1D-Convolution	64 filters, 3×1 kernel	SeLU, Batch Normalization
10	1D-Max-Pooling	2×1 pool	None
11	Flatten	N/A	None
12	Fully-Connected	128 neurons	SeLU
13	Dropout	Probability = 0.25	None
13	Fully-Connected	64 neurons	SeLU
14	Fully-Connected	10 neurons	Softmax

Figure 1: CNN architecture for an individual network

The *arctan2* function from NumPy is used instead of the standard mathematical function $tan^{-1}(\cdot)$ because the former has a range of $(-\pi, \pi]$, while the latter has a range of $(-\frac{\pi}{2}, \frac{\pi}{2})$ – thus, removing sign ambiguity.

3 Model Architecture

The model used for predicting the signal modulation type was a bagging of convolutional neural networks (CNNs). The convolutional layers capture spatial information, create feature maps, and reduce the dimensionality of the data, while the fully-connected layers help capture non-linearities in the data. The network was constructed using Keras and utilizes the Adam optimizer (with a learning rate of 0.0025 and batch size of 32) and the categorical cross-entropy loss function. As the natural complement for using the softmax function in the output layer, the predictions were outputted as one-hot encodings. Ten of these networks were "bagged" so as to reduce the variance of the overall model. In Keras, this involved averaging the outputs of the three individual networks.

4 Prediction

During training, each example matrix $\hat{\mathbf{X}}_i$ in $\hat{\mathbf{X}}_{trn}$ of size 256×4 is treated independently, thus their labels are predicted independently. Conversely, during testing, recall that each example $\mathbf{X}_i$ in $\mathbf{X}_{tst}$ of size 1024×4 was split into four examples of size 256×4 in $\hat{\mathbf{X}}_{tst}$: $\hat{\mathbf{X}}_{i,1,tst}$, $\hat{\mathbf{X}}_{i,2,tst}$, $\hat{\mathbf{X}}_{i,3,tst}$, $\hat{\mathbf{X}}_{i,4,tst}$. The label $l \in \mathcal{L}$ – where

$\mathcal{L}$ is the set of modulation types – of test example $\mathbf{X}_i$ is predicted using the four subsamples as follows:

$$\hat{y}_{\mathbf{X}_{i,tst}} = \underset{l \in \mathcal{L}}{\operatorname{argmax}} \prod_{j=1}^{4} \mathbb{P}(l|\mathbf{X}_{i,j,tst}) \tag{5}$$

where the values of $\mathbb{P}(l|\mathbf{X}_{i,tst})$, $\forall l \in \mathcal{L}$, are given by the output of the softmax activation function in the output layer of the CNN. This model achieved 61.820% accuracy on the testing set.

5 Other Approaches

I also attempted a wide variety of other approaches to this problem. In addition to the final features I decided on, I experimented with the following features:

- discrete Fourier transform coefficients and associated frequency and power values
- statistical analyses of the discrete wavelet transform approximate and detail coefficients
- low-pass filtering of the original signals
- images of the plots of the original signals
- principal components of I/Q data derived from PCA

Throughout the project, I also experimented with the following algorithms and models: random forests; decision trees; AdaBoost; gradient-boosted trees; support vector machines; k-nearest neighbors; and convolutional, long short-term memory, fully-connected deep neural networks (CLDNNs).

6 Conclusion

6.1 Bottlenecks

I found that large amount of noise in the dataset posed a large bottleneck. Because of this noise, models in which I used engineered features (e.g. statistical analyses of DWT coefficients) tended to greatly underperform as compared to deep learning approaches. Also, the high dimensionality of the dataset made training require significant amounts of time, which prompted me to focus on developing models and approaches that reduced the dimensionality, such as those that utilized pooling layers.

6.2 Learning Outcomes

I learned that there are times where domain knowledge-based feature engineering is no match for deep learning, especially with noisy data. The power of machine learning often comes from the ability of algorithms to decipher what is and what is not important in the data on their own – without expert feature engineering. I believe that by tackling a problem as broad and open-ended as this one, I was able to learn a lot about how to iterate through the model selection process. Given that I tried so many different algorithms and architectures, I feel that I have become fluent in using both Keras and sklearn and that I also better understand the material from the course.