Project Presentation

Enhancing Neural Networks with hyperparameters selection

Aleksandr Algazinov

Higher School of Economics

June 28, 2023

NATIONAL RESEARCH
UNIVERSITY

Abstract

Motivation

Deep Learning is one of the several areas of AI that are getting the most attention. So, it is important to understand, what caused this interest, and what is behind these rather complex models.

Research topic

Training neural networks is a long and complicated process. Therefore, it is necessary to understand, how to do it quickly and efficiently, and in which situations the use of Deep Learning generally makes sense.

What was done

We conducted a theoretical analysis, where we discussed, when using Deep Learning methods is the best or single option, and showed this with examples. In addition, we have identified several advises and rules that can be used to simplify the optimization of neural networks.

Introduction

When Deep Learning is relevant?

- 1. Big data
- 2. Specific tasks ML does not handle (embeddings, object recognition)
- 3. Complex problems where DL shows better performance than ML
- 4. Uncommon data structures (sounds, images)

Why to use Deep Learning now?

- 1. Availability of big data
- 2. Hardware (GPU, horizontal parallelization)
- 3. Software (PyTorch, Tensorflow)

Briefly about neural networks

Neural networks can be represented as a linear regression with nonlinear transformations:

$$\hat{y} = w_0 + x^T w \longrightarrow \hat{y} = g(w_0 + x^T w),$$

where g - activation function. Examples:
Sigmoid: $g(z) = \frac{1}{1+e^{-z}}$; TanH: $g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$; ReLU: $g(z) = max(0, z)$.
So, we apply different transformations to our input to get the output and the loss:

$$x \xrightarrow{w_1} z_1 \xrightarrow{w_2} \hat{y} \longrightarrow J(w).$$

Finally, we can compute gradients with backpropagation technique and use methods such as Adam to optimize a loss function:

$$\frac{\partial J(w)}{\partial w_1} = \frac{\partial J(w)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_1} \frac{\partial z_1}{\partial w_1}.$$

Features and problems of optimizing neural networks

Problems:

- 1. In general, the problem is not convex $\Rightarrow$ saddle points and local minimas.
- 2. Since we use a chain rule, gradients can become too large (exploding) or too small (vanishing).
- 3. The cost and the result of the process is very dependent on the choice of hyperparameters.

Solutions:

- 1. By using methods, such as SGD or Momentum, we introduce randomness and prevent gradient from being 0.
- 2. Use another activation function, reinitialize weights, or change the architecture of the network.
- 3. Use special libraries or find common patterns and advises to make proper choices.

Embeddings

What is an embedding?

A mapping of an entity to a fixed-length vector. So, we describe an object as a d-dimensional vector, assuming it can be explained by d aspects.

What is its value?

Machine Learning algorithms cannot work directly with complex data structures (like images or sounds). Using Deep Learning, we can convert them into objects that algorithms know how to work with.

Practical examples, where they are used

- 1. NLP (convert texts into vectors)
- 2. Reduce the dimension of the dataset
- 3. Recommend new items to users

Graph analysis (algorithm)

One of the methods to construct embeddings on graphs is called Node2Vec. We map nodes to fixed-length vectors: $\text{ENC}(v_i) = z_i$, estimate a conditional probability of moving from one node to another ($p(v_j|v_i)$), using random walk, and want to find embeddings, such that $p(v_j|v_i) \approx \frac{e^{z_j^T z_i}}{\sum_k e^{z_k^T z_i}} = \text{DEC}(z_i, z_j)$. This results in the optimization problem:

$$\sum_{v_i, v_j \in V} \ell\Big(\text{DEC}[\text{ENC}(v_i), \text{ENC}(v_j)], s_g(v_i, v_j)\Big) \longrightarrow \min_{\text{ENC,DEC}}.$$

As a loss function, we will use:

$$\ell = \sum_{(v_i, v_j)} -log(\text{DEC}(z_i, z_j)).$$

Graph analysis (example)

Problem and method

We have a graph of subjects that are taught in the first year of various HSE faculties (125 observations, 11 faculties, and 57 subjects). To be able to solve Machine Learning tasks with this data, we used built-in Python node2vec library to create embeddings.

Application 1: Recommendations

We looked at the most similar nodes to Faculty of Electronics and Mechanics. We identified that Faculty of CS and Faculty of Mathematics are similar to it, and that there are options to add new subjects or substitute existing ones.

Application 2: Classification

We can fit embeddings to AI models. So, we did a binary classification problem to show it works. There were 3 models to identify whether it is a popular subject or faculty, or otherwise.

Text classification (overview)

We want to predict country of the wine given its description (10 classes).
Naive Bayes: Given $\{x_i\}^D$ features and C classes, we can write a problem density (assuming that features affect the probabilities independently):

$$p(x|y = c, \theta) = \prod_{j=1}^{D} p(x_j|y = c, \theta_{jc}).$$

We want to optimize $p(y|x)$, so we apply Bayes rule:

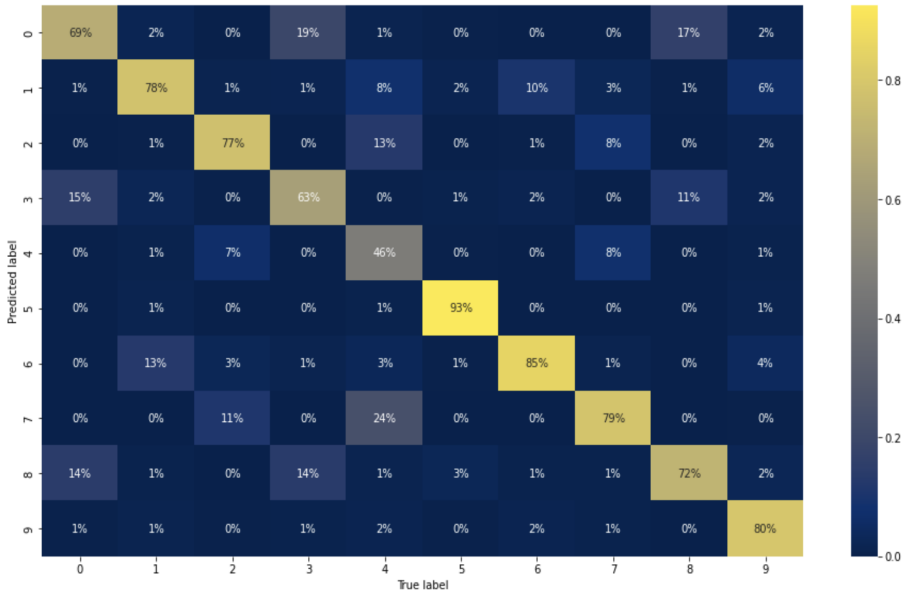
$$\text{argmax } p(y|x) = \text{argmax } p(x|y)p(y).$$

Problems: a strong assumption, ignoring text properties.
Solution: use a pretrained neural network, which looks at joint probabilities, overcomes the curse of dimensionality, and takes into account semantic similarities and differences between words.

Text classification (comparison of algorithms)

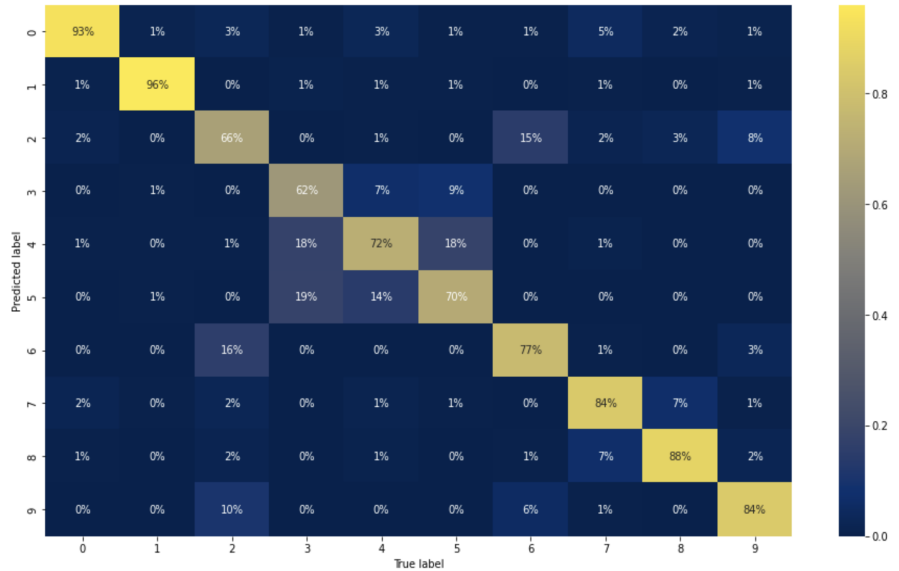
Accuracy is : 0.7414131501472031

	precision	recall	f1-score	support
Argentina	0.62	0.69	0.65	590
Australia	0.69	0.78	0.73	567
Austria	0.77	0.77	0.77	643
Chile	0.66	0.63	0.65	615
France	0.73	0.46	0.57	624
Italy	0.96	0.93	0.94	626
New Zealand	0.76	0.85	0.80	579
Portugal	0.69	0.79	0.73	628
Spain	0.65	0.72	0.68	591
US	0.92	0.80	0.85	651
accuracy			0.74	6114
macro avg	0.74	0.74	0.74	6114
weighted avg	0.75	0.74	0.74	6114



	precision	recall	f1-score	support
0	0.98	0.93	0.95	6209
1	0.96	0.96	0.96	2334
2	0.85	0.66	0.75	2056
3	0.82	0.62	0.70	877
4	0.57	0.72	0.64	589
5	0.58	0.70	0.64	569
6	0.54	0.77	0.64	554
7	0.64	0.84	0.73	482
8	0.70	0.88	0.78	350
9	0.50	0.84	0.62	315
accuracy			0.85	14335
macro avg	0.71	0.79	0.74	14335
weighted avg	0.87	0.85	0.85	14335

[72]: Text(123.0, 0.5, 'Predicted label')



Hyperparameters and my recommendations

- Number of neurons and hidden layers: this is probably the most important hyperparameter, and the choice depends on the situation.
- Activation and loss functions: use ReLU for hidden layers; loss function and function for the output layers are chosen based on problem.
- Optimizer: Adam is probably the best option, but some researchers say that in several situations SGD or RMSProp can be better.
- Batch size and number of epochs: consider the shape of the data while choosing batch size. Number of epochs can be controlled by early stopping.
- Learning rate: this is an important hyperparameter, so try several and see which is the best.
- Drop-out rate (rate of neurons that we will keep): the more you are concerned with overfitting, the less it should be.

Conclusions

- We found out why neural networks are so popular, and why all the conditions for their further development are met.
- We discussed the key problems with their optimization, as well as possible solutions.
- We looked at the concept of embeddings, and also gave an example of a task that cannot be solved without using them.
- We solved the problem of text classification, explaining in detail why the methods of Deep Learning outperform Machine Learning.
- We explained why hyperparameters play a key role in optimizing neural networks, and also identified several patterns and rules that will help speed up the process of choosing them.

Thank you for your attention!