Deep Learning in Data Science - Assignment 3 - Bonus Tasks

Fernando García Sanz

April 25, 2020

Abstract

The scope of this assignment consists in building and training a k-layer network that implements batch normalization and has multiple outputs. This network is used to classify the images contained in the CIFAR-10 [1] dataset. The network is trained using mini-batch gradient descent over a cost function which computes the cross-entropy loss of the classifier applied to the labelled training data and an L_2 regularization term over the weight matrix. Bonus tasks can be found in this document.

1 Exhaustive random search for regularization

Do a more exhaustive random search to find good values for the amount of regularization.

The network architecture and configuration for this improvement have been as follows:

- Architecture: [50, 50, 10]
- Configuration: η_{min} = 1e-5, η_{max} = 1e-1, $\lambda = \lambda_{search}$, $n_s = 5 * 45000/n_{batch}$

The amount of values employed in the coarse search of lambda has been increased from 10 to 40. After training with those 40 values the results obtained have been the following:

- Best accuracies in validation: [0.5406, 0.5404, 0.5394]
- Best lambdas in validation ($\log_{10}$ scale): [-2.26577879, -2.30251003, -2.30937357]

Using the best results to perform a fine search of other 40 lambda values, new results are obtained:

- Best accuracies in validation: [0.5406, 0.5406, 0.5404]
- Best lambdas in validation ($\log_{10}$ scale): [-2.26577879, -2.28956407, -2.30251003]

As can be seen, the best accuracies obtained almost do not improve those obtained in the previous search (some are repeated with different lambda values).

Finally, using the lambda value that provides the best accuracy, -2.26577879, the network has been trained for 3 cycles, providing a final accuracy over the test data of 53.66%. Applying the same configuration but only using 10 different lambdas during both searches returns similar results. Therefore, performing this exhaustive search does not make sense in this case, due to the number of computational resources employed in contrast to the outcomes obtained.

2 Thorough search of a good network architecture

Do a more thorough search to find a good network architecture. Does making the network deeper improve performance?

Trying some modifications of the architecture has made possible improving the performance. Different configurations have been tested, each of a different layer size and network deepness, and

also employing batch normalization, cyclical learning rates, lambda coarse-to-fine search and *He* initialization to obtain the best results possible.

The parameters used have been the following:

η_{min} = 1e-5, η_{max} = 1e-1, $\lambda = \lambda_{search}$, $n_s = 5 * 45000/n_{batch}$

- Configuration 1: [100, 50, 50, 10]
  - Best accuracies in validation: [0.552, 0.5502, 0.5496]
  - Best lambdas in validation ($\log_{10}$ scale): [-2.19345518, -2.03882999, -2.20067643]
  - Improved accuracies in validation: [0.5544, 0.5536, 0.5534]
  - Improved lambdas in validation ($\log_{10}$ scale): [-2.28872494, -2.22056151, -2.24405414]
  - Accuracy over test data: 54.55%
- Configuration 2: [50, 50, 50, 50, 50, 20, 10]
  - Best accuracies in validation: [0.5294, 0.5254, 0.521]
  - Best lambdas in validation ($\log_{10}$ scale): [-2.39125665, -2.01646904, -2.74493539]
  - Improved accuracies in validation: [0.5364, 0.5328, 0.5294]
  - Improved lambdas in validation ($\log_{10}$ scale): [-2.34402411, -2.17240126, -2.39125665]
  - Accuracy over test data: 52.29%
- Configuration 3: [100, 100, 100, 50, 50, 20, 10]
  - Best accuracies in validation: [0.5576, 0.5504, 0.5472]
  - Best lambdas in validation ($\log_{10}$ scale): [-2.20326297, -2.3956336, -2.18315896]
  - Improved accuracies in validation: [0.5582, 0.5576, 0.555]
  - Improved lambdas in validation ($\log_{10}$ scale): [-2.0277387, -2.20326297, -2.05676692]
  - Accuracy over test data: 54.86%

3 Dropout implementation

Apply dropout to your training if you have a high number of hidden nodes and you feel you need more regularization.

For this experiment the network structure and configurations have been the following:

- Architecture: [200, 50, 10]
- Configuration: η_{min} = 1e-5, η_{max} = 1e-1, $\lambda = \lambda_{search}$, $n_s = 5 * 45000/n_{batch}$

After training the model without regularization, the accuracy obtained over the test set has been equal to 53.88%. Nevertheless, when employing dropout over the same configuration, the result improves up to 56.27% of accuracy over the test set. This supposes an increment of a 2.39%, which is a quite considerable improvement.

This proves the efficacy of dropout as a method to avoid overfitting and improve the model generalization. In this case, as the number of neurons in the first layer was big (200), applying dropout makes more sense than in other configurations where the number of neurons that compose each layer is lower.

4 Data augmentation by jitter addition

Augment your training data by applying small random geometric and photometric jitter to the original training data. You can do this on the fly by applying a random jitter to each image in the mini-batch before doing the forward and backward pass.

Applying jitter to an image improves the generalization of the model when training.

The noise have being applied to one third of the pixels of each image. To do so, the pixel values have being slightly increased or decreased and the transformed pixels have been chosen randomly.

The following architecture and configuration have been used:

- Architecture: [50, 50, 10]
- Configuration: η_{min} = 1e-5, η_{max} = 1e-1, $\lambda = \lambda_{search}$, $n_s = 5 * 45000/n_{batch}$

The accuracy obtained over test set when applying jitter to the images has been equal to 53.59%, while when not applying jitter, it has been equal to 52.31%.

Therefore, it is clear that applying jitter has helped to enhance the model performance, obtaining an increment of 1.08% in the accuracy while keeping the configuration of both experiments identical.

References

[1] A. Krizhevsky. The cifar-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html. Accessed: 21-03-2020.