Due Date : March 9th, 2018

Instructions

- *For all questions, show your work !*
- *This part (practical) is to be done in teams of 2 or 3.*
- *Use a document preparation system such as LaTeX.*
- *Submit your answers electronically via the course studium page.*

1. **(35 points) Regularization : weight decay, early stopping, dropout, domain prior knowledge**

   Regularization is a technique used in machine learning that leverages additional information to solve an ill-posed problem or to improve generalization. We'll see a few instantiations of different principles of regularization below. All of these perspectives are orthogonal to each other, and can be combined to improve the model's performance. For this question, use the MNIST dataset.

   (a) *Early stopping and weight decay* : From the modeling point of view, we often want to have a mapping that is smooth with respect to the inputs, and this is often done by penalizing the magnitude of the parameters. Early stopping and weight decay are two such techniques.

   Instructions : For this part, all model architectures are 2 hidden layers with 800 units with ReLU activations. Use SGD with learning rate of 0.02, minibatch size of 64, and the Glorot Normal initialization. Train all models for 100 epochs.

   i. No regularisation

   ii. L2 regularisation

   Plot the L2 norm of all the parameters at each iteration (minibatch update) during the training of both models.

   For the L2 regularization specifically, we minimize the following objective :

   $$\sum_{i=1}^{n} l(x_i, y_i; \theta, b) + \frac{1}{2}\lambda||\theta||_2^2$$

   where $l(\cdot; \theta, b)$ is the cross entropy of the model parameterized by θ and b, θ is a vector representing all the weights and b represents the biases in the network, and n is the size of the *entire* training set. We set $\lambda = 2.5$ for this experiment. Note that you need to rescale the coefficient to adapt the loss for minibatch SGD.

   Instructions : Plot the error at the end of each epoch of training.

   (b) *Dropout* : It is sometimes hard to find a single hypothesis to perform well on a particular problem. This is often addressed by averaging multiple solutions from different learned models.

   Instructions : Use the same architecture as the MLP in Question (a). Apply dropout on the *last* hidden layer and train the model for 100 epochs. After training, evaluate it on the test set. For each data instance, use the following three schemes to get a prediction.

   i. Multiply 0.5 to the hidden layer before prediction.

   ii. Sample N dropout masks, and average the pre-softmax values, before applying softmax and making the prediction.

iii. Sample N dropout masks, and make a prediction for each one, and average the predictions.

Plot the graph for $N = 10, 20, 30, \ldots, 100$ (Part i. should be a straight line). What different kinds of ensemble methods are we applying for the different schemes above ?

(c) *Convolutional Networks* : Finally, many techniques correspond to incorporating certain prior knowledge of the structure of the data into the parameterization of the model. Convolution operation, for example, is designed for visual imagery.

Instructions : Train a convolutional network on MNIST for 10 epochs. Plot the error at the end of each epoch for the model.

i. with batch-normalization.

ii. without batch-normalization.

Use the architecture mentioned here [1] for the CNN part.

2. **(65 points) Dogs vs. Cats Classification**
Dogs vs Cats is a Kaggle challenge for image classification. Using what you have learned from Question 1, make an attempt at training the best CNN classifier for this task. Use the script here [2] to download, extract and process the raw data to 64×64 images. You may also do your own cropping, but state the image sizes you are using. This question is meant to give you a chance to play with the methods seen in class and apply them to a real classification task. Show your efforts by recording what you have tried on the report.

(a) Describe the architecture (number of layers, filter sizes, pooling, etc.), and report the number of parameters. You can take inspiration from some modern deep neural network architectures such as the VGG networks [SZ14] to improve the performance.

(b) Plot the training error and validation error. You can use the optimization techniques you learned in the class, such as different optimizers or feature normalization, to accelerate training. In particular, try Batch Normalization.

(c) Compare different hyperparameter settings and report the final results of performance on test set. Aside from quantitative results, also include some visual analysis such as visualizing the feature maps or kernels, or showing examples where the images are (a) clearly misclassified and (b) where the classifier predicts around 50% on both classes. Explain your observation and/or suggest any improvements you think may help.

Références

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*, 2014.

1. `https://github.com/MaximumEntropy/welcome_tutorials/tree/pytorch/pytorch`
2. `https://github.com/CW-Huang/IFT6135H18_assignment/blob/master/download_cat_dog.py`