Homework 8, due March 2nd, 11:59pm

February 23, 2022

1. Download the files `cnntrain.zip` and `cnntest.zip` from Canvas, containing training and test patches of size 64×64 for the problem of predicting the resolution of an image. The first two digits in each file name represent the resolution y_i of the patch (between 10% and 96%).

The goal of this project is to train a regression CNN to predict the resolution. We will use the square loss functions on the training examples $(\mathbf{x}_i, y_i), i = 1, ..., n$:

$$S(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f_{\mathbf{w}}(\mathbf{x}_i))^2 + \lambda \|\mathbf{w}\|^2 \tag{1}$$

Besides the loss function, we will measure the R^2, defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

where $\hat{y}_i = f_{\mathbf{w}}(\mathbf{x}_i)$ and $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$.

Experiment with different CNN architectures to obtain a good result. One example of a CNN you could use contains five convolutional layers with stride 1 and no padding, the first four with filters of size 5×5 with or without holes (dilation), and the last layer of the appropriate size to obtain a 1×1 output. The first two convolutions have 16 filters, the next two have 32 filters, and the last has only one filter. The second and third convolutions could be followed by 2×2 max pooling with stride 2 respectively. The fourth convolution layer is followed by ReLU.

Try to use a GPU and CUDA for faster training.

a) Train a CNN for 100 epochs with momentum 0.9 using the square loss (1). Use the SGD optimizer with an appropriate learning rate and $\lambda = 0.0001$ (weight decay). Start with minibatch size 64 and double it every 20 epochs and try to obtain a good training R^2 (at least 0.9). You might also need to reduce the learning rate as training progresses. Show a plot of the loss function vs epoch number for the training set and the test set. Show another plot of the training and test R^2 vs epoch number. (4 points)

b) Repeat point a) with the same CNN architecture, but with a fixed minibatch of 1024 and a fixed learning rate. It's ok if you cannot get a training R^2 of at least 0.9 in this case. (3 points)

c) Report the CNN architecture in a table, where each row describes one layer, including the layer description, size and number of parameters, and the last row containing the total number of parameters. (2 points)

d) Plot with two different colors the train and test residuals $r_i = y_i - \hat{y}_i$ vs y_i for the model obtained at a). (1 point)