Learning rate Choice using Radial Basis Functions

Ettore Celozzi
E-mail address
ettore.celozzi@stud.unifi.it

Luca Ciabini
E-mail address
luca.ciabini1@stud.unifi.it

1. Introduction

The aim of the project is to use the radial basis functions in order to find the best Learning Rate in a neaural network for the recognition of handwritten digits with a cross entropy loss function.
The goal for the network is: take an input image (28x28 pixels) of a handwritten single digit (0-9) and classify the image as the appropriate digit.

2. The network

Using Keras we create a 3 layer network:

- The first layer have 512 neurons and take as input features vector from the dataset MNIST that contains handwritten and labeled digits. The activation function is a Relu and has a dropout of 0.2.
- The hidden layer has the same number of neurons, the same activation function and the same dropout of the first layer.
- The last layer has, clearly, 10 neurons and a softmax activation function.

Once the model is created and the weights are initialized they are saved in the disk. In this way we can be sure that the weight are always the same for each evaluation of the model with a different learning rate.
The optimizer used for the model is the Adam while the loss function we decide to use is a $sparse\_categorical\_crossentropy$.

3. Radial Basis Function

Radial Basis function(Rbf) is a powerful tool to interpolate a set of points $(x_1 \dots x_k)$. The interpolation function is calculated from the subsequent formula:

$$s(x^i) = \sum_{j=1}^{k} \lambda_j \varphi(\|x^i - y^j\|) \tag{1}$$

where

- λ_j: are coefficients;
- $\varphi(\|x^i - x^j\|)$: basis function;
- y^j: $j-th$ center of radial basis function. Often this value is x^j, that is each point in the set.

Since we need a good surrogate model ($s(x)$) of expensive function $f(x)$, the RBF were the best way to get it. However the former formula 1 does not guarantee to have always a solution and another polynomial is needed.
The choice of basis function φ can avoid this requirement, in fact choosing a **gaussian** basis function, formula 1 always admits solution without any extra polynomial.
The form of gaussian basis function is:

$$\varphi(r) = exp(-\gamma r^2)$$

Equivalent formulation of 1, that involve matrices is:

$$s(x) = \phi\lambda \tag{2}$$

$$\phi_{ij} = \varphi(\|x^i - y^j\|) \tag{3}$$

3.1. RBF implementation

To implement RBF interpolation we created the class RBF. This class has various attribute:

- X: the set of points in logarithm scale (to avoid instability) to interpolate ;
- F: the value of the function evaluated in $x_i \in X$;
- $multipliers$: the coefficients λ_i of linear combination;
- phi: the matrix generated from i-th point calculated in the j-th center;
- $epsilon$: the constant of gaussian RBF

Through the function $interpolate()$, is possible to interpolate all the points in X. This function in fact resolve the liner system 2 to retrieve the multipliers λ.

4. Surrogate function optimization

The next step is to find the new convenient point in which evaluate the expensive function, is optimize the model retrieved from interpolation. The first idea of minimize the function $s(x)$ leads to **stalling**, and **locality**. The first means that the same point is founded in different iterations and second that the model become too local.
So the idea is to assume $\hat{f}$ to be an aspiration level and chose the corresponding point as the one that minimize the bumpiness of the function obtained from the interpolation of this new point and those former. This new point is added to the set and another iteration with a new surrogate function begin.
The new point is the solution of the following optimization problem:

$$\min_{\hat{x}} Bump(\hat{x}) \tag{4}$$

$$\text{where} \quad Bump(\hat{x}) = (\hat{f} - s(\hat{x}))^2 g(\hat{x}) \tag{5}$$

$$\text{and} \quad g(\hat{x}) = \frac{det\left[\phi\right]}{det\begin{bmatrix} \phi & \phi_{k+1} \\ \phi_{k+1}^T & 0 \end{bmatrix}} \tag{6}$$

4.1. Implementation

To resolve the problem 4 we have to define some values:

- $\hat{f}$: is found minimizing the interpolant through the function $minimizeRBF()$ or can be $-\infty$ to make an exploration pass;
- $s(xcap)$: the value of surrogate function in $\hat{x}$;
- $g(xcap)$: the function $g(xcap)$ calculate determinant and divide the results like in 6.

The function $newxGivenf(fvalue)$ get the estimate value $\hat{f}$ $(fvalue)$ and through **L-BFGS-B** minimize the bumpiness.
To avoid the optimization becoming too local, the minimization is done starting from a set of point (from -6 to -0.3). The result of each minimization is stored in an array and the point returned (the new point) is the min value among all the minimization results.
The minimization is performed thanks to $minimize()$ function of **scipy** [1] python library.

5. RBF interpolation

The first three learning rate are chosen randomly in a range between 10^{-6} and 0.5. Once these values are chosen the model is evaluated and the values of the loss are then used to create the interpolating function.

After that the program start looping until the number iteration reach the value of the variable $numberOfIterations$. In each iteration we call the function $newxGivenf()$ to get the new learning rate obtained minimizing the bumpiness and we convert it back to non-logarithmic scale (since inside the RBF we use logarithm scale to avoid instability). Furthermore every 3 iteration we pass to this function a $True$ value in order to make an exploration pass.
Then the model is evaluated with the new learning rate, the list containing all the x and y values are updated and we start again.
At the end the best learning rate found is given back and the rbf is plotted.

[1] https://docs.scipy.org/doc/scipy/reference/