test

gard.pavels

October 2021

1 Classification

1.1 Setting the NN up for classification

As briefly mentioned in the introduction, our implemented neural network will also be used in a classification context on the Wisconsin breast cancer data set. This dataset consists of 30 features computed from a sample of a breast mass taken from 569 individuals. The dataset also consists of a diagnosis attribute describing whether the current individual has benign or malignant cancer; in other words if the tumor in question is spreading or not.

The overall goal for the classification will be to predict diagnosis with great accuracy on unseen data, both by the neural network and the logistic regression model.

1.2 Implementing Logistic regression

To further validate the fitness of our implemented neural network, we also implemented a logistic regression model. These models are exceptionally well suited for cases where the goal is to assign observations to discrete classes. At the core of this regression model is the output of the probability of the observed data belonging to the class in question.

For a binary classification problem, as the Wisconsin breast cancer data, and 2 arbitrary parameters, the probabilites can be formulated as the sigmoid function, given as:

$$p(y_i = 1|x_i, \beta) = \frac{\exp(\beta_0 + \beta_1 x_i)}{1 + \exp(\beta_0 + \beta_1 x_i)},$$

$$p(y_i = 0|x_i, \beta) = 1 - p(y_i = 1|x_i, \beta)$$

where y_i is defined as the binary target data. For this specific case, we use a dataset with 30 features. The computation of probabilites can therefore be formulated as:

$$p(y_i = 1|x_i, \beta) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i + ... + \beta_{30} x_i)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i + ... + \beta_{30} x_i)},$$

$$p(y_i = 0|x_i, \beta) = 1 - p(y_i = 1|x_i, \beta),$$

Furthermore, we want to establish a cost function which will produce a convex plot. This is crucial, as a non-convex plot will create problems when trying to optimize the parameters using stochastic gradient descent. We need to ensure that any local minimizer is also a global minimizer[ref week38.ipynb]. To achieve this, we will opt for using cross-entropy, defined as:

$$\mathcal{C}(\beta) = -\sum_{i=1}^{n} \left(y_i(\beta_0 + \beta_1 x_i + ... + \beta_{30} x_i) - \log\left(1 + \exp\left(\beta_0 + \beta_1 x_i + ... + \beta_{30} x_i\right)\right)\right).$$

Our aim is to minimize this cost functions with respect to all parameters β for all n observations:

$$\frac{\partial \mathcal{C}(\beta)}{\partial \beta_0} = -\sum_{i=1}^{n} \left(y_i - \frac{\exp\left(\beta_0 + \beta_1 x_i + ... + \beta_{30} x_i\right)}{1 + \exp\left(\beta_0 + \beta_1 x_i + ... + \beta_{30} x_i\right)}\right),$$

.....

$$\frac{\partial \mathcal{C}(\beta)}{\partial \beta_{30}} = -\sum_{i=1}^{n} \left(y_i - \frac{\exp\left(\beta_0 + \beta_1 x_i + ... + \beta_{30} x_i\right)}{1 + \exp\left(\beta_0 + \beta_1 x_i + ... + \beta_{30} x_i\right)}\right),$$

For the Wisconsin breast cancer data set, we will define a target vector y, consisiting of the binary diagnostic data, a design matrix X, and a vector p consisting of the probabilites of each observations, produced by the above mentioned sigmoid function.

The first derivative of the cost function can then be formulated as:

$$\frac{\partial \mathcal{C}(\beta)}{\partial \beta} = -X^T\left(y - p\right).$$

We have chosen to implement logisitc regression with both stochastic gradient descent and Newton Raphson´s method. Algorithm XX describes how SGD was implemented. Here we have introduced a learning rate η and a regularization parameter λ. In this algorithm, all β parameters will be updated after each iteration through a mini-batch.

Another approach for logistic regression is to solve using Newton Raphson´s method. This approach forces us to introduce a term with second derivatives:

$$\frac{\partial^2 \mathcal{C}(\beta)}{\partial \beta \partial \beta^T} = X^T W X.$$

Here, matrix $W = p(y_i|x_i, \beta)(1 - p(y_i|x_i, \beta)$, is computed with the p values computed in same manner as in the algorithm for stochastic gradient descent.

As demonstrated by Algorithm X and algorithm Y, there is several distinct differences between the two implmentations. While Newton Raphson´s method only iterates through the epochs, the stochastic gradient descent will have the additional iterations through the batches. This might lead to the impression that the latter is computationally more complex than the first. This is however

Data: Design Matrix (X), target array (t) and initial guess at predictors θ
Result: Estimated value of the parameters β
for *epoch in number of epochs* **do**
  for *batch in number of batches* **do**
    $x_i \leftarrow X[batch]$
    $t_i \leftarrow t[batch]$
    $p \leftarrow probabilites(\mathrm{x}_i, \beta)$;
    Compute $\frac{\partial \mathcal{C}(\beta)}{\partial \beta}$;
    $\beta \leftarrow \beta * \lambda^2$;
    $\beta \leftarrow \beta - \eta * \frac{\partial \mathcal{C}(\beta)}{\partial \beta}$;
  end
end
Algorithm 1: Logistic Regression with Stochastic Gradient Descent

Data: Design Matrix (X), target array (t) and initial guess at predictors θ
Result: Estimated value of the parameters β
for *epoch in number of epochs* **do**
  $p \leftarrow probabilites(\mathrm{x}_i, \beta)$;
  $W = \mathrm{p}(\mathrm{y}_i|x_i, \beta)(1 - p(y_i|x_i, \beta)$;
  $hessian \leftarrow X^T W X$;
  $\beta \leftarrow \beta$ - $(X^T W X)^{-1} \times \frac{\partial \mathcal{C}(\beta)}{\partial \beta}$
end
Algorithm 2: Logistic Regression with Newton Raphson´s method

not the case, as the inversion of $k \times k$ matrix has a complexity of $O(k^3)$, which has to be carried out each iteration as the parameters change with every update[ref deeplearningbook]. For cases with a large number of parameters, the exponentially incraeasing computational burden imposed often makes SGD the preferred solver.

In this project, we have implemented and tested both.

Common for both algorithms is the approach in how we compute the accuracy. With all the epochs done, β has reached its final estimation, based on the training data. We compute the probabilites of the observed data in the test set X_{test} with:

$$p(y_{test}|X_{test}, \beta) = \frac{1}{1 + \exp -(\beta X_{test})}$$

The values are then treshholded, assigning all probabilites > 0.5 to class 1 and all < 0.5 to 0. The accuracy for n samples is then simply calculated by:

$$Accuracy = \frac{\sum_{i=1}^{n} I(t_i = y_i)}{n},$$