

6.867 Problem Set 3

November 10, 2015

Neural Networks

In some cases, we might want to use a neural network rather than another algorithm, such as SVM or logistic regression, in order to be somewhat more agnostic about the parametric form of the data-generating process. In this case, we investigate the use of neural networks, first on some toy training data sets, and subsequently on a subset of the popular MNIST dataset.

Let us first briefly review the way that neural network works. A neural network consists of a variable number of “hidden layers“. A weight matrix maps each of your data points to a value for each node in the hidden layer. We then evaluate some “activation function“ on each node value (typically a sigmoid function or tanh function) to get a new set of “inputs“ for the next layer of the neural network. We continue this process until we finally arrive at our output predictions. A schematic sketch of a neural net can be found below.

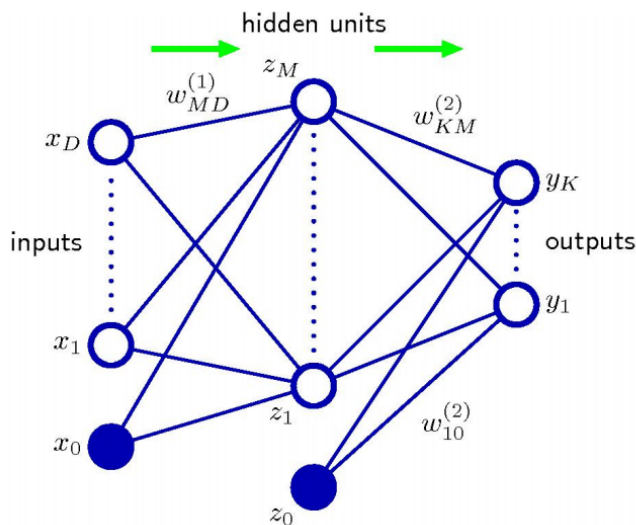


Figure 1: A schematic diagram of a neural network (taken from the homework assignment document)

Alternatively, a neural network may be specified as follows:

$$a_n^{(1)} = \sum_d w_{dn}^{(1)} x_d \quad (1)$$

$$z_n^{(1)} = \sigma_1(a_n^{(1)}) \quad (2)$$

$$a_k^{(2)} = \sum_n w_{nk}^{(2)} z_n \quad (3)$$

$$z_k^{(2)} = \sigma_2(a_k^{(2)}) \quad (4)$$

The most trivial neural network has only one output layer with one node, and no hidden layers. This would correspond to vanilla logistic regression.

For the purposes of this paper, we will try to solve simple multi-class prediction problems using a neural network with one hidden layer. In order to determine the weight matrices that give us the best predictive accuracy, we will specify a loss function and perform (batch or stochastic) gradient descent. We specify our objective function as the regularized negative log-likelihood function:

$$J(w) = l(w) + \lambda(\|w^{(1)}\|_F^2 + \|w^{(2)}\|_F^2) \quad (5)$$

where $l(w)$ is

$$l(w) = \sum_{i=1}^N \sum_{k=1}^K [-y_k^{(i)} \log(h_k(x^{(i)}, w)) - (1 - y_k^{(i)}) \log(1 - (h_k(x^{(i)}, w)))]. \quad (6)$$

In order to calculate the gradient as a function of w_1 and w_2 , we use the two gradient calculation expressions below:

$$\frac{\partial J(w)}{\partial w_{kj}^{(2)}} = \sum_{i=1}^N z_j \left(\sigma(a_k^{(2)}) - y_k^{(i)} \right) + 2\lambda w_{kj}^{(2)} \quad (7)$$

and

$$\frac{\partial J(w)}{\partial w_{jd}^{(1)}} = \sum_{i=1}^N x_i \sum_{k=1}^K \left(\sigma(a_k^{(2)}) - y_k^{(i)} \right) w_{kj}^{(2)} \left(\frac{1}{1 + e^{-a_{jd}^{(1)}}} \right) \left(1 - \frac{1}{1 + e^{-a_{jd}^{(1)}}} \right) + 2\lambda w_{jd}^{(1)}. \quad (8)$$

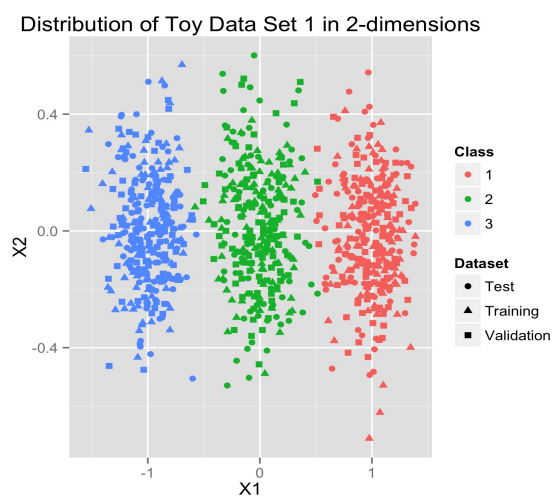
Let's now see how a neural network performs on two different toy data sets. We vary both the number of hidden nodes and the regularization parameter in our objective function, and use both full gradient descent and batch gradient descent. The two algorithms perform comparably - in the case of toy data set number 2, we find that the full gradient descent algorithm does a slightly better job at finding the optimal values of our parameter than stochastic gradient descent (classification error of 6.7% as opposed to 9%). Our changes in parameters have a much larger effect on the performance for toy dataset 2 than for our features in toy dataset 1. This is likely due to the lack of an easy way to separate examples of different classes in dataset 2.

Table 1: Neural Network Classification Error Rates With Various Parameters on Toy Datasets

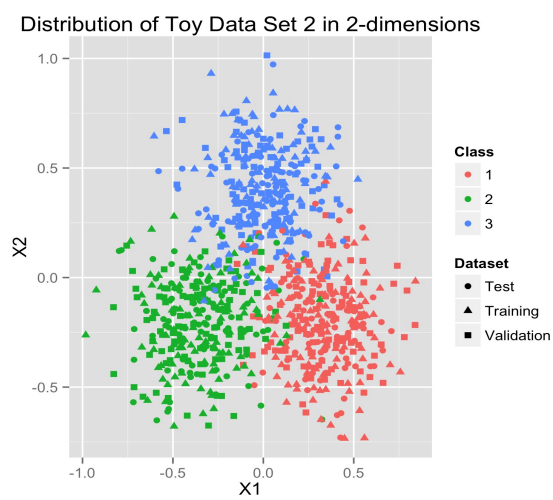
Dataset	Method	λ	$n_1 = 1$	$n_1 = 3$	$n_1 = 5$	$n_1 = 7$	$n_1 = 9$
Toy Dataset 1 (Training)	full gradient descent	0	0.007	0	0	0	0
Toy Dataset 1 (Training)	full gradient descent	1e-5	0.007	0	0	0	0
Toy Dataset 1 (Training)	full gradient descent	1e-3	0.093	0.003	0.003	0.003	0.003
Toy Dataset 1 (Training)	full gradient descent	1e-1	0.667	0.667	0.667	0.667	0.667
Toy Dataset 1 (Validation)	full gradient descent	0	0.01	0.003	0.003	0.003	0.003
Toy Dataset 1 (Validation)	full gradient descent	1e-5	0.01	0.003	0.003	0.003	0.003
Toy Dataset 1 (Validation)	full gradient descent	1e-3	0.063	0.003	0.003	0.003	0.007
Toy Dataset 1 (Validation)	full gradient descent	1e-1	0.667	0.667	0.667	0.667	0.667
Toy Dataset 1 (Training)	SGD	0	0.003	0.003	0.003	0.003	0.003
Toy Dataset 1 (Training)	SGD	1e-5	0.043	0	0.003	0.003	0
Toy Dataset 1 (Training)	SGD	1e-3	0.117	0	0.003	0.003	0.003
Toy Dataset 1 (Training)	SGD	1e-1	0.667	0.667	0.667	0.667	0.667
Toy Dataset 1 (Validation)	SGD	0	0.003	0.003	0.01	0.007	0.003
Toy Dataset 1 (Validation)	SGD	1e-5	0.033	0.007	0.003	0.007	0.003
Toy Dataset 1 (Validation)	SGD	1e-3	0.097	0.013	0.013	0.013	0.013
Toy Dataset 1 (Validation)	SGD	1e-1	0.667	0.667	0.667	0.667	0.667
Toy Dataset 2 (Training)	full gradient descent	0	0.36	0.063	0.047	0.05	0.027
Toy Dataset 2 (Training)	full gradient descent	1e-5	0.397	0.063	0.067	0.063	0.067
Toy Dataset 2 (Training)	full gradient descent	1e-3	0.413	0.06	0.06	0.06	0.06
Toy Dataset 2 (Training)	full gradient descent	1e-1	0.667	0.667	0.667	0.667	0.667
Toy Dataset 2 (Validation)	full gradient descent	0	0.343	0.07	0.093	0.1	0.107
Toy Dataset 2 (Validation)	full gradient descent	1e-5	0.39	0.073	0.07	0.073	0.073
Toy Dataset 2 (Validation)	full gradient descent	1e-3	0.38	0.067	0.067	0.067	0.067
Toy Dataset 2 (Validation)	full gradient descent	1e-1	0.667	0.667	0.667	0.667	0.667
Toy Dataset 2 (Training)	SGD	0	0.363	0.073	0.053	0.05	0.06
Toy Dataset 2 (Training)	SGD	1e-5	0.363	0.07	0.067	0.057	0.067
Toy Dataset 2 (Training)	SGD	1e-3	0.363	0.07	0.07	0.07	0.07
Toy Dataset 2 (Training)	SGD	1e-1	0.667	0.667	0.667	0.667	0.667
Toy Dataset 2 (Validation)	SGD	0	0.373	0.083	0.11	0.103	0.113
Toy Dataset 2 (Validation)	SGD	1e-5	0.373	0.087	0.077	0.08	0.09
Toy Dataset 2 (Validation)	SGD	1e-3	0.377	0.097	0.097	0.093	0.093
Toy Dataset 2 (Validation)	SGD	1e-1	0.667	0.667	0.667	0.667	0.667

Table 2: Neural Net Classification Error Rates on MNIST dataset for different choices of parameters

Dataset	lr	method	λ	$n_1 = 50$	$n_1 = 100$	$n_1 = 150$
Training	.1	full gradient descent	0	0	0	0
Training	.1	full gradient descent	1e-3	0.007	0.005	0.006
Training	.1	full gradient descent	1e-2	0.069	0.069	0.074
Training	.1	full gradient descent	1e-1	0.812	0.812	0.812
Validation	.1	full gradient descent	0	0.068	0.078	0.062
Validation	.1	full gradient descent	1e-3	0.064	0.07	0.062
Validation	.1	full gradient descent	1e-2	0.072	0.078	0.078
Validation	.1	full gradient descent	1e-1	0.822	0.822	0.822
Training	.1	SGD	0	0.112	0.109	0.106
Training	.1	SGD	1e-3	0.109	0.119	0.114
Training	.1	SGD	1e-2	0.13	0.132	0.131
Training	.1	SGD	1e-1	0.683	0.703	0.78
Validation	.1	SGD	0	0.09	0.1	0.086
Validation	.1	SGD	1e-3	0.092	0.09	0.098
Validation	.1	SGD	1e-2	0.122	0.122	0.12
Validation	.1	SGD	1e-1	0.68	0.698	0.762
Training	.01	full gradient descent	0	0.084	0.064	0.057
Training	.01	full gradient descent	1e-3	0.076	0.067	0.056
Training	.01	full gradient descent	1e-2	0.094	0.087	0.081
Training	.01	full gradient descent	1e-1	0.663	0.66	0.66
Validation	.01	full gradient descent	0	0.09	0.076	0.08
Validation	.01	full gradient descent	1e-3	0.102	0.074	0.07
Validation	.01	full gradient descent	1e-2	0.088	0.08	0.076
Validation	.01	full gradient descent	1e-1	0.674	0.672	0.672
Training	.01	SGD	0	0.14	0.122	0.115
Training	.01	SGD	1e-3	0.153	0.132	0.124
Training	.01	SGD	1e-2	0.153	0.152	0.138
Training	.01	SGD	1e-1	0.629	0.545	0.511
Validation	.01	SGD	0	0.14	0.116	0.118
Validation	.01	SGD	1e-3	0.136	0.126	0.108
Validation	.01	SGD	1e-2	0.128	0.142	0.124
Validation	.01	SGD	1e-1	0.622	0.548	0.504



Toy Dataset #1



Toy Dataset #2

Figure 1: Distributions of different classes in X_1 , X_2 for toy datasets