CS7643: Deep Learning
Spring 2019
Homework 0, Solutions

Alexis DUROCHER (903336265 adurocher3)

January 16, 2019

1 Softmax Classifier

In this homework I implemented a Softmax classifier using no machine learning libraries. This classifier will be trained on the CIFAR10 dataset.

This project included the implementations of a vectorized loss function, gradient computation and the Stochastic Gradient Descent optimization algorithm.

```
vectorized loss: 2.334888e+00 computed in 0.616283s
loss: 2.334888
sanity check: 2.302585
numerical: 1.119079 analytic: 1.119079, relative error: 3.024530e-08
numerical: 0.862233 analytic: 0.862233, relative error: 1.129072e-07
numerical: 1.717056 analytic: 1.717056, relative error: 1.883843e-09
numerical: -0.026962 analytic: -0.026962, relative error: 9.196089e-07
numerical: 1.159507 analytic: 1.159506, relative error: 7.469083e-08
numerical: 0.812624 analytic: 0.812624, relative error: 5.848146e-08
numerical: -0.095026 analytic: -0.095026, relative error: 1.231718e-07
numerical: -2.028350 analytic: -2.028350, relative error: 6.513904e-09
numerical: 1.452112 analytic: 1.452112, relative error: 5.690599e-08
numerical: 4.474878 analytic: 4.474878, relative error: 8.192315e-09
```

Figure 1: Gradient check from cell 3 output

1. Softmax regression uses the cross-entropy loss L and a regularization term R to penalize high-valued weights. The final loss L_{tot} is the addition of:

   $L = \frac{-1}{N} \sum_i^N log(p_i^{y_i})$ and $R(W) = \sum_k \sum_l W_{k,l}^2$

   Note that the regularization term is multiplied by a hyper-parameter constant θ *(equal to 0 for the gradient check as computed in Figure 1).*

   The gradient $\nabla_W L_{tot} = \nabla_W L + \theta \nabla_W R$.

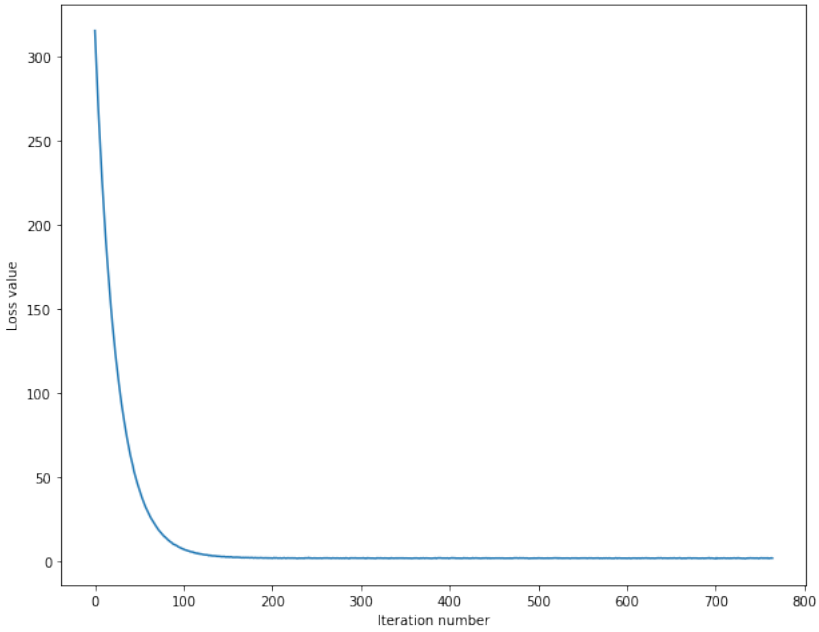


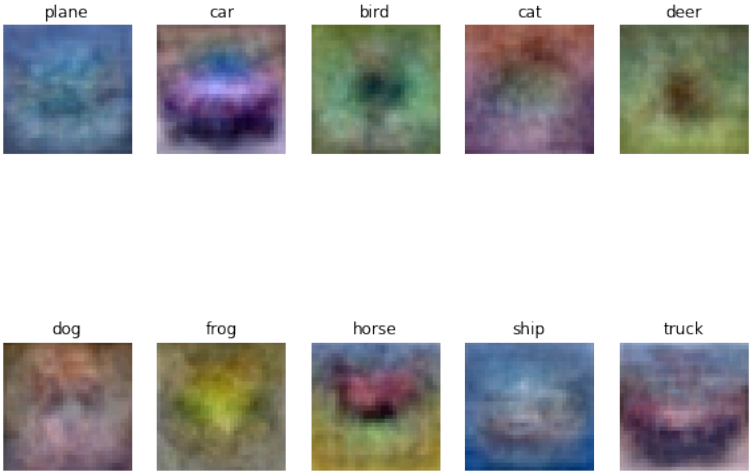
Figure 2: Softmax loss minimization

2. The gradient $\nabla_W L_{tot} = \nabla_W L + \nabla_W R$. The gradient is used in the Stochastic Gradient Descent (SGD) to update the matrix of parameters W in order to minimize iteratively the loss function (see Figure 2). Since the softmax loss is a convex function, it converges toward a global minimum, given appropriate hyper parameters.

   Softmax computations may lead to numerical instabilities because of the normalization by exponential numbers. To cope with this issue, I restrained the z_i values by adding a constant term $\alpha_i = -max(z_i) + min(z_i)$ to the z_i values. Note that adding constant terms to the z_i values don't impact the softmax outputs p_i *(ref [1] for the proof of mathematical consistency)*.

   To fine-tune the model, I used the validation set to find the best regularization parameter θ and learning rate lr via a grid search over those two hyper parameters. I noticed that the distribution of the validation and test datasets were not very correlated since better accuracy (fine-tuned parameters) on the Training dataset didn't necessarily yield better test accuracy nor visuliaztion. However, the grid search still helped reducing the possibilities for those parameters and find a correct trade-off. Also, I reduced the batch size from 200 to 64 in order to proportionally increase the number of SGD iterations per epochs ($N/batchsize$).

   The final accuracy was around 30 %. This result is poor but we could expect the linear classifier to perform poorly regarding the very likely non-linearity of this classification problem, and the high-dimension of our data.



3. Once the classifier is trained, the weights are not updated anymore. We can visualize each weight "layers" (i.e each row W_j) per class j.

   It is interesting to notice shape and color patterns in the weights layer corresponding to the class their are related to. For the ship class for instance, we can observe an average blue and grey/white color corresponding to an average ship shape surrounded by water. For the frog class, weights have an average green/brown color and a pattern similar to an average frog's shape can be detected. Similar observations can be drawn for each other weight layers and their respective class.

   This observation is a proof that the classifier learned the appropriate patterns to recognize the different class w.r.t to the training dataset. However, the low accuracy of the model indicates that this model clearly over-fitted on the training samples and thus, didn't generalize to edge cases *(i.e a bird surronded by water could be predicted as a plane or ship using this classifier)*.

References

[1] Standfor. *CS231n Convolutional Neural Networks for Visual Recognition* `http://cs231n.github.io/linear-classify/`