EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATH. - NATURWISS. FAKULTÄT
FACHBEREICH INFORMATIK
KOGNITIVE SYSTEME · PROF. A. ZELL

Deep Neural Networks

Assignment 5

Assignment due by: 12.06.2018, Discussions on: 19.06.2018

Question 1 Softmax (2 + 2 + 2 + 2 + 3 points)

The last weeks assignment we used logistic regression to perform binary classification. If we want to extend this to more than two classes, we can use softmax function defined by:

$$\boldsymbol{S}(\boldsymbol{a}) : \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_N \end{bmatrix} \rightarrow \begin{bmatrix} S_1 \\ S_2 \\ \cdots \\ S_N \end{bmatrix} \text{ with } S_i = \frac{e^{a_i}}{\sum_j e^{a_j}}$$

The properties of softmax (all output values in the range $[0, 1]$ and sum up to 1) make it suitable for a probabilistic interpretation that's very useful in machine learning. In particular, in multiclass classification tasks, we often want to assign probabilities that our input belongs to one of a set of output classes. To train a model with this, we can minimize the cross-entropy $H(y, \boldsymbol{S}(\boldsymbol{a}))$ between the true distribution (which is just probability 1 on the correct class, 0 on all others) and the predicted probabilities $\boldsymbol{S}(\boldsymbol{a})$ (where $\boldsymbol{a}$ is the un-normalized output of our classifier, the preactivation). Thus we get the following loss function (for a single point of data):

$$\mathcal{L}(\boldsymbol{a}, y) = H(y, \boldsymbol{S}(\boldsymbol{a})) = -\sum_c \delta_{yc} \log(\boldsymbol{S}(\boldsymbol{a})_c) = -\log(\boldsymbol{S}(\boldsymbol{a})_y)$$

where δ_{yc} is the Kronecker delta which is 1 if $y = c$ and 0 otherwise.

(a) Compute $\nabla_{\boldsymbol{a}} \mathcal{L}$, i.e. the gradient of the output of $\mathcal{L}$ with respect to the preactivations $\begin{bmatrix} a_1 & a_2 & \cdots & a_N \end{bmatrix}$.

(b) Implement the softmax classification function (without cross-entropy) as a fully vectorized function. (For this part, (c), (d) and (e), use the provided python template file $softmax.py$)

(c) Implement the softmax cross-entropy loss $\mathcal{L}(\boldsymbol{a}, y)$ as a fully vectorized function.

(d) Implement the analytical gradient of the softmax cross-entropy loss from (a) as a fully vectorized function.

(e) Test your functions from (c) and (d) by comparing the analytical gradient to a numerical gradient at randomly chosen points. Besides, describe the difference between analytical gradient and numerical gradient with your own understanding through this case.

fully vectorized means the function should be able to handle both a single input, as well as multiple inputs at the same time (matrix of multiple $\boldsymbol{a}$, vector of multiple y). This is important for processing batches of input efficiently and will be needed in future assignments.

Note: Make sure you apply the numerical stabilization techniques from chapter 4.

Question 2 k Nearest Neighbors and Cross Validation (4 + 5 points)

In this assignment we will develop a Nearest Neighbor Classifier. This classifier has nothing to do with Convolutional Neural Networks and it is rarely used for image classification, but it will allow us to get an idea about the basic approach to an image classification problem, in our case the classic MNIST handwritten digit recognition task.

The kNN classifier consists of two stages:

- During training, the classifier takes the training data and simply remembers it
- During evaluation, kNN finds the k closest training points to the test point (usually by simple euclidean distance) and classifies it by the most common class among those k nearest neighbors.

An outline of the code for Question 2 and the MNIST dataset can be downloaded from ILIAS.

(a) Implement the k-Nearest Neighbors algorithm. Evaluate its accuracy on the test set for $k \in \{1, 3, 5, 15\}$

(b) Now ignore the test set and implement n-fold cross validation. Choose the k with the best accuracy from (a) and perform cross-validation for $n \in \{2, 3, 5, 8, 12\}$. Comment on the results.