Deep Structured Learning (IST, Fall 2019)

Homework 1

Instructor: André Martins and Vlad Niculae
TAs: Gonçalo Correia and Ben Peters

Deadline: Friday, October 11, 2019.

Please turn in the answers to the questions below in a PDF file, together with the code you implemented to solve them (when applicable). Please email your solutions in **electronic format** (a single zip file) with the subject "Homework 1" to:

`deep-structured-learning-instructors@googlegroups.com`

Hard copies will not be accepted.

Question 1

In this exercise, we will explore the ability of perceptrons and multilayer perceptrons to classify whether a vector belongs to a set. Concretely, the set we will consider is the set of points in $\mathbb{R}^2$ that lie inside the intersection of the non-negative orthant and the ℓ_1-ball.

The d-dimensional ℓ_1-ball is defined as:

$$\mathcal{B}^d := \left\{ (x_1, \dots, x_n) \in \mathbb{R}^d \;\middle|\; \sum_{n=1}^{d} |x_n| \le 1 \right\}, \tag{1}$$

and the non-negative orthant is the set of all vectors with non-negative coordinates, i.e.,

$$\mathbb{R}^d_+ := \left\{ (x_1, \dots, x_n \in \mathbb{R}^d \;\middle|\; x_i \ge 0 \text{ for } 1 \le i \le n \right\} \tag{2}$$

The set we are interested in is their intesection:

$$\mathcal{B}^d_+ := \mathcal{B}^d \cap \mathbb{R}^d_+. \tag{3}$$

In prose, a vector is in $\mathcal{B}^d_+$ if all of its entries are nonnegative and together they sum to less than or equal to 1. In the following questions, we will consider the particular case where $d = 2$ and implement a classifier that returns a binary label $y \in \{0, 1\}$ for any point $\mathbf{x} \in \mathbb{R}^2$. The classifier should return

$$y = \begin{cases} 1 & \text{if } \mathbf{x} \in \mathcal{B}^2_+, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

As our nonlinearity, we will use the Heaviside step function $H: \mathbb{R} \to \{0, 1\}$,

$$H(z) = \begin{cases} 1 & \text{if } z \ge 0, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

1. (5 points) Can a linear model perfectly classify whether $\mathbf{x} \in \mathcal{B}_+^d$? Why or why not?

2. (5 points) The nonnegativity constraints from the definition of $\mathbb{R}_+^d$ state that $x_1 \geq 0$ and $x_2 \geq 0$. Exhibit weights $\mathbf{w} \in \mathbb{R}^2$ and bias $b \in \mathbb{R}$ for a single-layer perceptron, such that $H(\mathbf{w}^\top \mathbf{x} + b)$ returns

$$y = \begin{cases} 1 & \text{if } x_1 \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

   Hint: Every weight and bias value can be set to either 0 *or* 1*.*

3. (5 points) Exhibit weights $\mathbf{w} \in \mathbb{R}^2$ and bias $b \in \mathbb{R}$ for a single-layer perceptron that returns

$$y = \begin{cases} 1 & \text{if } x_1 + x_2 \leq 1, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

   Hint: All parameters can be set to ± 1*.*

4. (20 points) Design a two-layer perceptron that has the same behavior as the classifier in Equation 4.

   Hint: $\mathcal{B}_+^2$ *is defined by a set of constraints which all need to be satisfied:* $x_1 \geq 0$*,* $x_2 \geq 0$*, and* $x_1 + x_2 \leq 1$*. Try to design your MLP so that each unit in the hidden layer computes one of these constraints, and then design connections between the hidden layer and output so that the classifier only returns* 1 *if all constraints are met.*

5. (5 points) Suppose you wanted to generalize your classifier from the previous question from $\mathcal{B}_+^2$ to an arbitrary dimension $\mathcal{B}_+^d$. As you increase d, how many more hidden units are required?

Question 2

Optical character recognition with linear classifiers. In this exercise, you will implement a linear classifier for a simple image classification problem. **Please do not use any machine learning library such as `scikit-learn` or similar for this exercise; just plain linear algebra.**

Download the OCR dataset from `http://ai.stanford.edu/~btaskar/ocr`. This dataset contains binary image representations of 52,152 alphabetical characters `a`–`z` (the characters are grouped together to form English words, but this structure will be ignored in this exercise). The task is to take each image representation as input (with 16x8 pixels) and to predict as output the correct character in `a`–`z` (i.e., a multi-class classification problem with 26 classes). The dataset is organized into 10 folds: folds 0–7 are for training (41,679 examples), 8 is for validation (5,331 examples), and 9 is for testing (5,142 examples). The evaluation metric is the fraction of characters correctly classified.

Skeleton code For Questions 2 and 3, you are recommended but not required to use the skeleton script hw1.py, which has been provided to you on the course webpage. In order to use it, make sure it is located in the same directory as the `letter.data` file from the OCR dataset. The script requires python 3, numpy, and matplotlib.

1. In the first part of the exercise, we will use as a feature representation the binary pixel values.

   (a) (5 points) Do you think this is a good choice of feature representation? Justify.
   (b) (10 points) Implement the `update_weights` method of the `Perceptron` class in `hw1.py`. Then train 20 epochs of the perceptron on the training set and report its performance on the validation and test set. Plot the accuracies as a function of the epoch number. You can do this with the command

```
python hw1.py perceptron
```

2. Let us now do some feature engineering.
   (a) (10 points) Can you think of a better feature representation? Come up with one and train the perceptron there. You can implement your feature representation in the function `custom_features` and then run

```
python hw1.py perceptron -custom_features
```

   Suggestion: instead of individual pixel binary values $\phi_i(\boldsymbol{x}) = x_i$ (where i indexes a pixel position), use as features all pixel pairwise combinations, $\phi_{ij}(\boldsymbol{x}) = x_i x_j$.
   (b) (10 points) Repeat the same exercise using logistic regression instead (without regularization), using stochastic gradient descent as your training algorithm. Set a fixed learning rate $\eta = 0.001$. This can be solved by implementing the `update_weights` method in the `LogisticRegression` class.
   (c) (5 points (bonus)) Add ℓ_2 regularization with a suitable regularization constant. What do you observe?

Question 3

Optical character recognition with a neural network. In the previous exercise, you might have noticed that feature engineering can be tedious. Now, you will implement a multi-layer perceptron (a feed-forward neural network) again using as input the original feature representation (i.e. simple independent pixel values).

1. (5 points) Explain why multi-layer perceptrons can learn internal representations and avoid manual feature engineering.

2. (20 points) **Without using any neural network toolkit,** implement a multi-layer perceptron with a single hidden layer to solve this problem, including the gradient backpropagation algorithm which is needed to train the model. Use your favorite activation function. Don't forget to tune all your hyperparameters.

3. (5 points (bonus)) Repeat the exercise above with multiple hidden layers and comment on the results.