Deep Structured Learning (IST, Fall 2018)

Homework 1

Instructor: André Martins
TAs: Vlad Niculae and Erick Fonseca

Deadline: Wednesday, October 10, 2018.

Please turn in the answers to the questions below 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

Multi-layer perceptron with quadratic activations. In this exercise, we will consider a feed-forward neural network with a single hidden layer and a quadratic activation function, $g(z) = z^2$. We will see under some assumptions, this choice of activation, unlike other popular activation functions such as tanh, sigmoid, or relu, does not get us far from a simple linear model.

We assume a univariate regression task, where the predicted output $\widehat{y} \in \mathbb{R}$ is given by $\widehat{y} = \boldsymbol{v}^\top \boldsymbol{h}$, where $\boldsymbol{h} \in \mathbb{R}^K$ are internal representations, given by $\boldsymbol{h} = \boldsymbol{g}(\boldsymbol{W}\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^D$ is a vector of input variables, and $\Theta = (\boldsymbol{W}, \boldsymbol{v}) \in \mathbb{R}^{K \times D} \times \mathbb{R}^K$ are the model parameters.

1. (10 points) Show that we can write $\boldsymbol{h} = \boldsymbol{A}_\Theta \boldsymbol{\phi}(\boldsymbol{x})$ for a certain feature transformation $\boldsymbol{\phi} : \mathbb{R}^D \to \mathbb{R}^{\frac{D(D+1)}{2}}$ independent of Θ and $\boldsymbol{A}_\Theta \in \mathbb{R}^{K \times \frac{D(D+1)}{2}}$. That is, $\boldsymbol{h}$ is a **linear transformation** of $\boldsymbol{\phi}(\boldsymbol{x})$. Determine the mapping $\boldsymbol{\phi}$ and the matrix $\boldsymbol{A}_\Theta$.

2. (5 points) Based on the previous claim, show that $\widehat{y}$ is also a linear transformation of $\boldsymbol{\phi}(\boldsymbol{x})$, i.e., we can write $\widehat{y}(\boldsymbol{x}; \boldsymbol{c}_\Theta) = \boldsymbol{c}_\Theta^\top \boldsymbol{\phi}(\boldsymbol{x})$ for some $\boldsymbol{c}_\Theta \in \mathbb{R}^{\frac{D(D+1)}{2}}$. Does this mean this is a linear model in terms of the original parameters Θ?

3. (10 points) Assume $K \geq D$. Show that any such $\boldsymbol{c}_\Theta \in \mathbb{R}^{\frac{D(D+1)}{2}}$ can be obtained by some choice of the original parameters $\Theta = (\boldsymbol{W}, \boldsymbol{v})$. That is, **we can directly parametrize the model with $\boldsymbol{c}_\Theta$ instead of Θ without losing any expressive power.** Does this mean this is a linear model in terms of $\boldsymbol{c}_\Theta$?

4. (5 points) Suppose we are given training data $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ with $N > \frac{D(D+1)}{2}$, and that we want to minimize the squared loss

$$L(\boldsymbol{c}_\Theta; \mathcal{D}) = \frac{1}{2} \sum_{n=1}^{N} (\widehat{y}_n(\boldsymbol{x}_n; \boldsymbol{c}_\Theta) - y_n)^2.$$

Can we find a closed form solution $\widehat{\boldsymbol{c}}_\Theta$? Is this a global or a local optimum?

5. (5 points) Find a way of recovering the original parameters $\Theta = (\boldsymbol{W}, \boldsymbol{v})$ (not necessarily unique) given the solution $\widehat{\boldsymbol{c}}_\Theta$ (hint: use orthogonal decomposition). Show that such Θ is a **global minimizer** of

$$L(\Theta; \mathcal{D}) = \frac{1}{2} \sum_{n=1}^{N} (\widehat{y}_n(\boldsymbol{x}_n; \Theta) - y_n)^2,$$

even though this objective function is non-convex in Θ. Does this happen for any activation function g?

6. (5 points) Does the above hold when $K < D$? Justify.

7. (5 points (bonus)) Determine the set of achievable parameters $\boldsymbol{c}_\Theta$ when $K < D$.

Question 2

Optical character recognition with linear classifiers. In this exercise, you will implement a linear classifier from scratch 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: please use folds 0–7 for training (41,679 examples), 8 for validation (5,331 examples), and 9 for testing (5,142 examples). The evaluation metric is the fraction of characters correctly classified.

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) 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.

2. Let us now do some feature engineering.
   (a) (5 points) Can you think of a better feature representation? Come up with one and train the perceptron there. 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, $\phi_{ij}(\boldsymbol{x}) = x_i x_j$. Does this choice corresponds to any kernel function?
   (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 $\eta = 0.001$. What did you need to change in your perceptron code? If you used a multi-class support vector machine instead, what else would you need to change?
   (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) using again 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.