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 $\boldsymbol{A}_\Theta$.

   Solution: Let $\boldsymbol{w}_i$ be the ith column of $\boldsymbol{W}$. We have

$$h_i = (\boldsymbol{w}_i^\top \boldsymbol{x})^2 = \boldsymbol{x}^\top \boldsymbol{w}_i \boldsymbol{w}_i^\top \boldsymbol{x} = \langle\!\langle \boldsymbol{w}_i \boldsymbol{w}_i^\top, \boldsymbol{x}\boldsymbol{x}^\top \rangle\!\rangle, \tag{1}$$

   where $\langle\!\langle\rangle\!\rangle$ denotes the Frobenius inner product $\langle\!\langle \boldsymbol{A}, \boldsymbol{B} \rangle\!\rangle = \mathrm{vec}(\boldsymbol{A})^\top \mathrm{vec}(\boldsymbol{B}) = \mathrm{Tr}(\boldsymbol{A}^\top \boldsymbol{B})$. Let

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} x_1^2 \\ \vdots \\ x_D^2 \\ 2x_1x_2 \\ \vdots \\ 2x_{D-1}x_D \end{bmatrix}, \quad \boldsymbol{a}_i = \begin{bmatrix} w_{i1}^2 \\ \vdots \\ w_{iD}^2 \\ w_{i1}w_{i2} \\ \vdots \\ w_{i(D-1)}w_{iD} \end{bmatrix}, \quad \boldsymbol{A}_\Theta = \begin{bmatrix} \boldsymbol{a}_1^\top \\ \vdots \\ \boldsymbol{a}_K^\top \end{bmatrix}. \tag{2}$$

   We then have $h_i = \boldsymbol{a}_i^\top \boldsymbol{\phi}(\boldsymbol{x})$ and $\boldsymbol{h} = \boldsymbol{A}_\Theta \boldsymbol{\phi}(\boldsymbol{x})$.

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?

   Solution: We have $\widehat{y} = \boldsymbol{v}^\top \boldsymbol{h} = \boldsymbol{v}^\top \boldsymbol{A}_\Theta \boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{c}_\Theta^\top \boldsymbol{\phi}(\boldsymbol{x})$, with $\boldsymbol{c}_\Theta = \boldsymbol{A}_\Theta^\top \boldsymbol{v}$. The resulting model is **not** linear in Θ, because $\boldsymbol{c}_\Theta$ is not a linear function of Θ: Although $\boldsymbol{c}_\Theta$ is a linear combination of rows of $\boldsymbol{A}_\Theta$, all entries of $\boldsymbol{A}_\Theta$ are **quadratic** – not linear – in terms of $\boldsymbol{W}$.

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 $\boldsymbol{c}_\Theta$?

   Solution: From above, we have $\boldsymbol{v}^\top \boldsymbol{h} = \sum_{i=1}^K v_i h_i = \sum_{i=1}^K v_i \langle\!\langle \boldsymbol{w}_i \boldsymbol{w}_i^\top, \boldsymbol{x}\boldsymbol{x}^\top \rangle\!\rangle = \langle\!\langle \sum_{i=1}^K v_i \boldsymbol{w}_i \boldsymbol{w}_i^\top, \boldsymbol{x}\boldsymbol{x}^\top \rangle\!\rangle = \langle\!\langle \boldsymbol{C}, \boldsymbol{x}\boldsymbol{x}^\top \rangle\!\rangle$, where $\boldsymbol{C} = \boldsymbol{W}^\top \mathrm{Diag}(\boldsymbol{v}) \boldsymbol{W}$ is a symmetric matrix. Note that we have $\boldsymbol{c}_\Theta = \mathrm{vech}(\boldsymbol{C})$, where $\mathrm{vech}(\boldsymbol{C})$ denotes the half-vectorization of $\boldsymbol{C}$, which forms a vector by collecting the elements in the lower triangular part of $\boldsymbol{C}$. We have

   $$\left\{ \boldsymbol{C} = \boldsymbol{W}^\top \mathrm{Diag}(\boldsymbol{v}) \boldsymbol{W} \mid \boldsymbol{W} \in \mathbb{R}^{K \times D}, \boldsymbol{v} \in \mathbb{R}^K \right\} = \left\{ \boldsymbol{C} \in \mathbb{R}^{D \times D} \mid \mathrm{rank}(\boldsymbol{C}) \leq K \right\}, \qquad (3)$$

   hence if $K \geq D$ this set equals $\mathbb{R}^{D \times D}$, from which we can obtain any $\boldsymbol{c}_\Theta \in \mathbb{R}^{\frac{D(D+1)}{2}}$ through the vech operation. Together with 1.2, this means that 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?

   Solution: Let $\boldsymbol{X} \in \mathbb{R}^{N \times \frac{D(D+1)}{2}}$ have $\boldsymbol{\phi}(\boldsymbol{x}_n)$ as rows, and define $\boldsymbol{y} = (y_1, \ldots, y_N)$. We want to minimize $\frac{1}{2}\|\boldsymbol{X}\boldsymbol{c}_\Theta - \boldsymbol{y}\|_F^2$ with respect to $\boldsymbol{c}_\Theta$. This is a least squares problem whose solution is $\widehat{\boldsymbol{c}}_\Theta = \boldsymbol{X}^+ \boldsymbol{y} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$. This is a global 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?

   Solution: Let us start by inverting the vech operation to obtain the symmetric matrix $\boldsymbol{C}$ from $\widehat{\boldsymbol{c}}_\Theta$. From the answer of question 1.3, we have $\boldsymbol{C} = \boldsymbol{W}^\top \mathrm{Diag}(\boldsymbol{v}) \boldsymbol{W}$. Let $\boldsymbol{C} = \boldsymbol{Q} \mathrm{Diag}(\boldsymbol{\lambda}) \boldsymbol{Q}^\top$ be an eigendecomposition of $\boldsymbol{C}$, where $\boldsymbol{\lambda}$ is the vector of eigenvalues and $\boldsymbol{Q}$ an orthogonal matrix with the corresponding eigenvectors as columns. Then we can "pad $\boldsymbol{Q}$ and $\boldsymbol{\lambda}$ with zeros" and take $\boldsymbol{W} = \begin{bmatrix} \boldsymbol{Q}^\top \\ \boldsymbol{0}_{K-D,D} \end{bmatrix}$ and $\boldsymbol{v} = \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{0}_{K-D} \end{bmatrix}$. This is a global minimizer (though not unique). This does not happen for any activation function—in general, it is very hard to obtain a global solution Θ for an activation such as tanh or sigmoid.

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

   Solution: No, since in that case a solution of the least squares problem $\widehat{\boldsymbol{c}}_\Theta$ may yield a matrix $\boldsymbol{C}$ with rank greater than K, from which one cannot recover a pair $\Theta = (\boldsymbol{W}, \boldsymbol{v}) \in \mathbb{R}^{K \times D} \times \mathbb{R}^K$.

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

   Solution: From the answer to 1.3, this set is $\left\{\mathrm{vech}(\boldsymbol{C}) \in \mathbb{R}^{\frac{D(D+1)}{2}} \,\middle|\, \mathrm{rank}(\boldsymbol{C}) \le K\right\}$.

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.

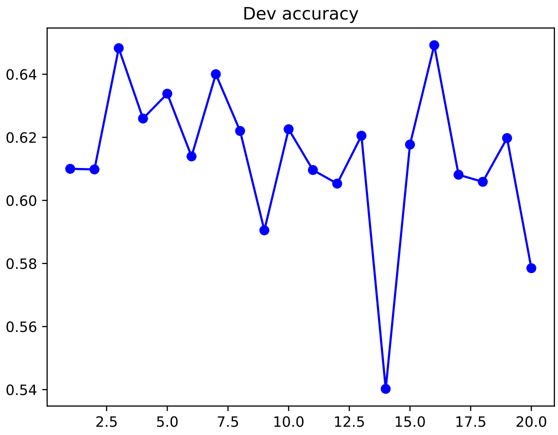
1. In the first part of the exercise, we will use as a feature representation the binary pixel values.

   (a) (5 points) Do you think this is a good choice of feature representation? Justify.

   Solution: Not for use with a linear classifier, since it is highly translation variant and does not exploit the fact that pixels are highly correlated.

   (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.

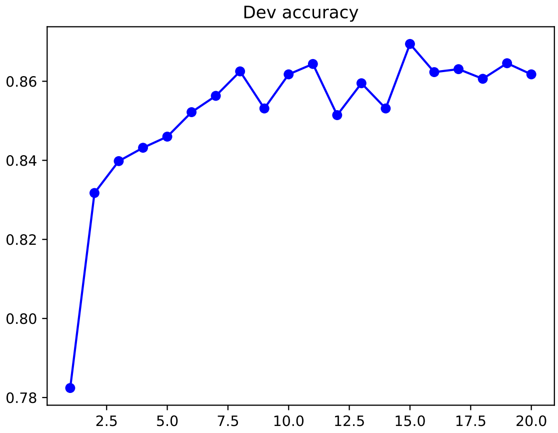
   Solution: Accuracies were 57.9% on the validation set and 57.0% on the test set.



2. Let us now do some feature engineering.

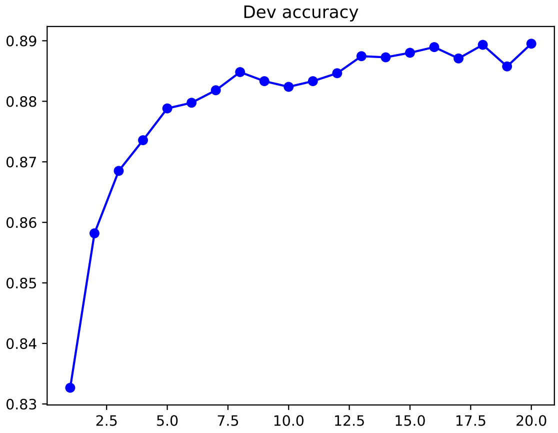
   (a) (10 points) Can you think of a better feature representation? Come up with one and train the perceptron there. Suggestion: instead of individual pixel binary values $\phi_i(\boldsymbol{x}) = x_i$ (where i indexes a pixel position), use as features all pixel pairwise combinations, $\phi_{ij}(\boldsymbol{x}) = x_i x_j$. Does this choice corresponds to any kernel function?

   Solution: It corresponds to a polynomial (quadratic) kernel. We have $k(\boldsymbol{x}, \boldsymbol{x}') = \sum_i \sum_j \phi_{ij}(\boldsymbol{x})\phi_{ij}(\boldsymbol{x}') = \sum_i \sum_j x_i x_j x'_i x'_j = \sum_i x_i x'_i \sum_j x_j x'_j = (\sum_i x_i x'_i)^2 = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^2$. With this feature representation we got accuracies of 86.2% on the validation set and 85.6% on the test set.



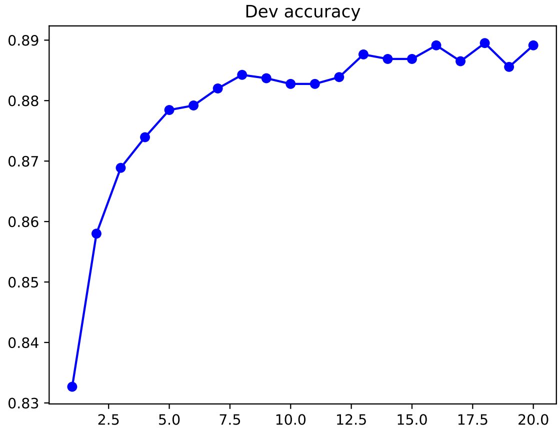
   (b) (10 points) Repeat the same exercise using logistic regression instead (without regularization), using stochastic gradient descent as your training algorithm. Set a fixed learning rate $\eta = 0.001$. What did you need to change in your perceptron code? If you used a multi-class support vector machine instead, what else would you need to change?

   Solution: With logistic regression with pairwise features we got accuracies of 89.0% on the validation set and 87.9% on the test set. Only the updates need to change.



   (c) (5 points (bonus)) Add ℓ_2 regularization with a suitable regularization constant. What do you observe?

   Solution: With $\lambda = 0.0001$ we got accuracies of 89.0% on the validation set and 88.0% on the test set. Other values of λ gave worse results. Regularization didn't make a big difference, arguably because the dataset was large enough that the model was not overfitting the data.



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.