CSCI 567, Fall 2018
Haipeng Luo

Homework #2 Written Assignments

Due: 11:59 pm, October 7, 2018

Instructions

Submission: Assignment submission will be via courses.uscden.net. By the submission date, there will be a folder set up in which you can submit your files. Please be sure to follow all instructions outlined here.

You can submit multiple times, but only *the last submission* counts. As a results, if you finish some problems, you might want to submit them first, and update later when you finish the rest. You are encouraged to do so. This way, if you forget to finish the homework on time or something happens (remember Murphy's Law), you still get credit for whatever you have turned in.

Problem sets must be typewritten or neatly handwritten when submitted. In both cases, your submission must be a single PDF. It is strongly recommended that you typeset with LaTeX. There are many free integrated LaTeX editors that are convenient to use (e.g Overleaf, ShareLaTeX). Choose the one(s) you like the most. This tutorial Getting to Grips with LaTeX is a good start if you do not know how to use LaTeX yet.

Please also follow the rules below:

- The file should be named as `firstname_lastname_USCID.pdf` e.g., `Don_Quijote_de_la_Mancha_8675309045.pdf`).
- Do not have any spaces in your file name when uploading it.
- Please include your name and USC ID in the header of the report as well.

Collaboration: You may discuss with your classmates. However, you need to write your own solutions and submit separately. Also in your report, you need to list with whom you have discussed for each problem. Please consult the syllabus for what is and is not acceptable collaboration. Review the rules on academic conduct in the syllabus: a single instance of plagiarism can adversely affect you significantly more than you could stand to gain.

Notes on notation:

- Unless stated otherwise, scalars are denoted by small letter in normal font, vectors are denoted by small letters in bold font and matrices are denoted by capital letters in bold font.
- $\|.\|$ means L2-norm unless specified otherwise i.e. $\|.\| = \|.\|_2$

Misc. Note that some texts are clickable hyper-links where you can find more information in the linked pages.

Problem 1 Neural networks

[Recommended maximum time spent: 1 hour]

In the lecture, we have talked about error-backpropagation, a way to compute partial derivatives (or gradients) w.r.t the parameters of a neural network. We have also mentioned that optimization is challenging

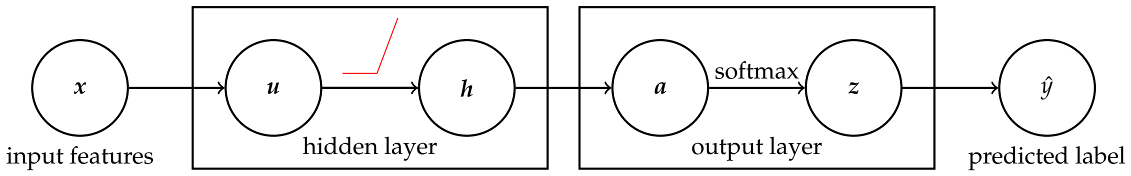

Figure 1: A diagram of a 1-hidden layer neural net. *The edges mean mathematical operations, and the circles mean variables.* Generally we call the combination of a linear (or affine) operation and a nonlinear operation (like element-wise sigmoid or the rectified linear unit (relu) operation as in eq. (3)) as a hidden layer. Note the two slight differences compared to the diagram used in the lecture : 1) one circle represents a vector and thus an array of neurons here and 2) the activation operations are also explicitly represented as edges here.

and nonlinearity is important for neural networks. In this question, you are going to (Q1.1) practice error-backpropagation, (Q1.2) investigate how initialization affects optimization, and (Q1.3) the importance of nonlinearity.

Specifically, you are given the following 1-hidden layer neural net for a K-class classification problem (see Fig. 1 for illustration and details), and $(\boldsymbol{x} \in \mathbb{R}^D, y \in \{1, 2, \cdots, K\})$ is a labeled instance,

$$\boldsymbol{x} \in \mathbb{R}^D \tag{1}$$

$$\boldsymbol{u} = \boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}, \quad \boldsymbol{W}^{(1)} \in \mathbb{R}^{M\times D} \text{ and } \boldsymbol{b}^{(1)} \in \mathbb{R}^M \tag{2}$$

$$\boldsymbol{h} = \max\{0, \boldsymbol{u}\} = \begin{bmatrix} \max\{0, u_1\} \\ \vdots \\ \max\{0, u_M\} \end{bmatrix} \tag{3}$$

$$\boldsymbol{a} = \boldsymbol{W}^{(2)}\boldsymbol{h} + \boldsymbol{b}^{(2)}, \quad \boldsymbol{W}^{(2)} \in \mathbb{R}^{K\times M} \text{ and } \boldsymbol{b}^{(2)} \in \mathbb{R}^K \tag{4}$$

$$\boldsymbol{z} = \begin{bmatrix} \frac{e^{a_1}}{\sum_k e^{a_k}} \\ \vdots \\ \frac{e^{a_K}}{\sum_k e^{a_k}} \end{bmatrix} \tag{5}$$

$$\hat{y} = \arg\max_k z_k. \tag{6}$$

For K-class classification problem, one popular loss function for training is the cross-entropy loss. Specifically we denote the cross-entropy loss with respect to the training example $(\boldsymbol{x}, y)$ by l:

$$l = -\ln(z_y) = \ln\left(1 + \sum_{k\neq y} e^{a_k - a_y}\right)$$

Note that one should look at l as a function of the parameters of the network, that is, $\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \boldsymbol{W}^{(2)}$ and $\boldsymbol{b}^{(2)}$.

Q1.1 Error Back-propagation Assume that you have computed $\boldsymbol{u}$, $\boldsymbol{h}$, $\boldsymbol{a}$, $\boldsymbol{z}$, given $(\boldsymbol{x}, \boldsymbol{y})$. Follow the four steps below to find out the derivatives of l with respect to all the four parameters $\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \boldsymbol{W}^{(2)}$ and $\boldsymbol{b}^{(2)}$. You are encouraged to use matrix/vector forms to simplify your answers. Note that we follow the convention that the derivative with respect to a variable is of the same dimension of that variable. For example, $\frac{\partial l}{\partial \boldsymbol{W}^{(1)}}$ is in $\mathbb{R}^{M\times D}$. (This is called the denominator layout.)

1. First express $\frac{\partial l}{\partial \boldsymbol{a}}$ in terms of $\boldsymbol{z}$ and y. You may find it convenient to use the notation $\boldsymbol{y} \in \mathbb{R}^K$ whose k-th coordinate is 1 if $k = y$ and 0 otherwise.

2. Then express $\frac{\partial l}{\partial \boldsymbol{W}^{(2)}}$ and $\frac{\partial l}{\partial \boldsymbol{b}^{(2)}}$ in terms of $\frac{\partial l}{\partial \boldsymbol{a}}$ and $\boldsymbol{h}$.

3. Next express $\frac{\partial l}{\partial \boldsymbol{u}}$ in terms of $\frac{\partial l}{\partial \boldsymbol{a}}$, $\boldsymbol{u}$, and $\boldsymbol{W}^{(2)}$. You will need to use the (sub)derivative of the ReLU function $\max\{0, u\}$ denoted by $H(u)$ and is 1 if $u > 0$ and 0 otherwise. Also, you may find it convenient to use the notation $\boldsymbol{H}(\boldsymbol{u}) \in \mathbb{R}^{M \times M}$ which stands for a diagonal matrix with $H(u_1), \ldots, H(u_M)$ on the diagonal.

4. Finally, express $\frac{\partial l}{\partial \boldsymbol{W}^{(1)}}$ and $\frac{\partial l}{\partial \boldsymbol{b}^{(1)}}$ in terms of $\frac{\partial l}{\partial \boldsymbol{u}}$ and $\boldsymbol{x}$.

Q1.2 Initialization Suppose we initialize $\boldsymbol{W}^{(1)}$, $\boldsymbol{W}^{(2)}$, $\boldsymbol{b}^{(1)}$ with zero matrices/vectors (i.e., matrices and vectors with all elements set to 0), please first verify that $\frac{\partial l}{\partial \boldsymbol{W}^{(1)}}, \frac{\partial l}{\partial \boldsymbol{W}^{(2)}}, \frac{\partial l}{\partial \boldsymbol{b}^{(1)}}$ are all zero matrices/vectors, irrespective of $\boldsymbol{x}$, $\boldsymbol{y}$ and the initialization of $\boldsymbol{b}^{(2)}$.

Now if we perform stochastic gradient descent for learning the neural network using a training set $\{(\boldsymbol{x}_i \in \mathbb{R}^D, \boldsymbol{y}_i \in \mathbb{R}^K)\}_{i=1}^N$, please explain with a concise mathematical statement (in one sentence) why no learning will happen on $\boldsymbol{W}^{(1)}$, $\boldsymbol{W}^{(2)}$, $\boldsymbol{b}^{(1)}$ (i.e., they will not change no matter how many iterations are run). Note that this will still be the case even with L2 regularization and momentum if the initial velocity vectors/matrices are set to zero.

Q1.3 Non-linearity As mentioned in the lecture, non-linearity is very important for neural networks. With non-linearity (e.g., eq. (3)), the neural network shown in Fig. 1 can bee seen as a nonlinear basis function $\boldsymbol{\phi}$ (i.e., $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{h}$) followed by a linear classifier f (i.e., $f(\boldsymbol{h}) = \hat{y}$).

Please show that, by removing the nonlinear operation in eq. (3) and setting eq. (4) to be $\boldsymbol{a} = \boldsymbol{W}^{(2)}\boldsymbol{u} + \boldsymbol{b}^{(2)}$, the resulting network is essentially a linear classifier. More specifically, you can now represent $\boldsymbol{a}$ as $\boldsymbol{U}\boldsymbol{x} + \boldsymbol{v}$, where $\boldsymbol{U} \in \mathbb{R}^{K \times D}$ and $\boldsymbol{v} \in \mathbb{R}^K$. Please write down the representation of $\boldsymbol{U}$ and $\boldsymbol{v}$ using $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{b}^{(1)}$, and $\boldsymbol{b}^{(2)}$.

Problem 2 Kernel methods

[Recommended maximum time spent: 1 hour]

In class, we studied kernel functions and their properties. Consider the following function:

$$k(\mathbf{x}, \mathbf{x}') = \begin{cases} 1, & \text{if } \mathbf{x} = \mathbf{x}' \\ 0, & \text{otherwise} \end{cases} \quad \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^D. \tag{7}$$

Q2.1 Prove that this is a valid kernel. You can apply the Mercer's theorem mentioned in the lecture and for simplicity assume that the N points $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ you pick are distinct (i.e., $\mathbf{x}_i \neq \mathbf{x}_j$ if $i \neq j$). (If you want, you can also prove without making this assumption with Mercer's theorem or even prove by using the definition of kernel functions.)

Q2.2 Suppose now you are given a training set $\{(\mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R})\}_{n=1}^N$ for a linear regression problem, where $\mathbf{x}_i \neq \mathbf{x}_j$ if $i \neq j$. Show that by using this kernel, the least square solution (with no regularization) will always lead to a total square loss of 0—meaning that all the training examples are *predicted accurately* by the least square solution.

Q2.3 Although the least square solution has 0 loss on the training set, it in fact does not generalize to the test data at all (that is, this algorithm completely overfits the training data). Specifically, show that for any unseen data point $\mathbf{x}$, that is, $\mathbf{x} \notin \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the prediction of the least square solution is always 0. (If you have worked out the feature mapping in Q2.1, which was optional, you will understand better why this happens.)