50.039 – Theory and Practice of Deep learning

Week 3

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.]

1 Hyperplanes

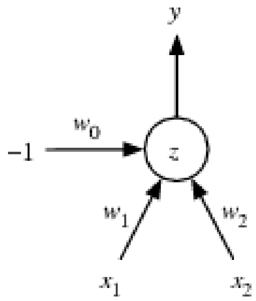
Suppose you have, in 3 dimensional space, the following dataset:
$D_n = \{(x_1 = (-5, 1, 3),\ y_1 = +1),\ \ (x_2 = (2, 2, -3),\ y_2 = -1)\}$.

Find one linear classifier $f(x) = wx + b$ (means its parameters), which predicts all points correctly.

Hint:
It can help to think of a hyperplane defined in terms of an orthogonal vector and a bias term. Get the orthogonal vector to the hyperplane right first, bias you can determine then afterwards by plugging in points, and solving for the bias.
For two points, what would be a good orthogonal vector for a hyperplane if you have two points which need to be separated??

2 Basic NN

The prelude to neural nets: classification with some activation function.
For this problem, we will consider the simple type of unit shown below.



The output of the unit $g(z)$ is computed as follows:

$$g(z) = \begin{cases} z & \text{if } |z| \leq 1 \\ sign(z) & \text{otherwise} \end{cases}$$

$$z = w_1 x_1 + w_2 x_2 - w_0$$

We can use this type of unit to classify our inputs by assigning any input for which the output is greater than or equal to 0 as positive, and for which the output is less than 0 to negative.
Given the four data points:
- Positive: $(0, 0)$, $(0, -1)$,
- Negative: $(1, 0)$, $(1, -1)$.
Choose a set of weights for this unit, so that the weights w_0, w_1, w_2 can separate these points.

3 Logistic Regression

1) In Page 6 of the lecture slides (`https://www.dropbox.com/s/fq5b1oxpr25t6hu/l06_logreg.pdf?dl=0`), we see that:

$$s(a) = \frac{\exp(a)}{1 + \exp(a)} == \frac{1}{\exp(-a) + 1}$$

Refer to Page 20 of the slides, please prove:

$$\frac{\partial log(s(a))}{\partial a} = 1 - s(a)$$

$$\frac{\partial log(1 - s(a))}{\partial a} = -s(a)$$

2) As shown in Pages 19-20 of the slides, we know that $h(x) = s(w \cdot x)$, and the objective function L is:

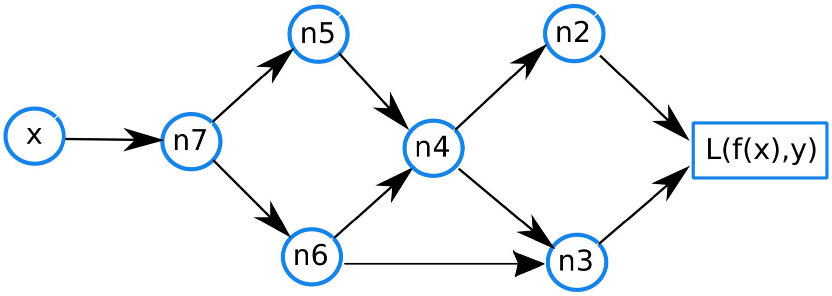
$$(-1) \cdot \sum_{i=1}^{n} y_i \log (h(x_i)) + (1 - y_i) \log (1 - h(x_i))$$

Prove that the gradient of L with respect to w will be:

$$\nabla_w L = \sum_{i=1}^{n} x_i(s(w \cdot x_i) - y_i) = \sum_{i=1}^{n} x_i(h(x_i) - y_i)$$

4 Back-Propagation

Compute the gradients of L with respect to all the neuron outputs (and also with respect to the input x), for the graph below. You are suggested to refer to Pages 10-20 of the slides about back-propagation.



5 Some Einsum

Which einsum notation is required to implement the following operations? Remember "einsum notation" is a pair:

$$indices_1, indices_2, indices_3, \ldots, indices_n - > indices_r, [t_1, t_2, t_3, \ldots, t_n]$$

Note: After you solve the following problems in this homework, you need to test your "einsum notations" in PyTorch with CoLab or other platforms by testing some examples, to see if your solution can really get the effects that you want. For this, you can refer to Sections 2.1 to 2.10 of this web page: `https://rockt.github.io/2018/04/30/einsum`, where some examples (i.e., the code blocks in black boxes) are used to test the effect of the "einsum notation".
You need to upload your testing codes as supplementary of this homework. The format of testing codes can be very flexible (.py and Jupyter notebook files are both acceptable). You can choose to upload a separate code file for each problem, or one file including all testing examples (python codes).

All in all, for this problem, you need to provide the "einsum notations" for the following operations in your homework. You also need to upload the codes for testing your "einsum notations".

- $$C_{j,k} = \sum_i A_{ijk} b_i$$

- $$A_{ik} = \sum_{j,l} A_{ijkl}$$

- Note: this is not the same as above. Note the change in index ordering

  $$A_{ki} = \sum_{j,l} A_{ijkl}$$

- $$C_i = \sum_{j,k} A_{ijk} A_{ijk}$$

- $$C = AG^\top B, A \in \mathbb{R}^{d\times e},\ 2 - tensor, G \in \mathbb{R}^{f\times e},\ 2 - tensor, B \in \mathbb{R}^{f\times l},\ 2 - tensor,$$

  Hint: What order will be the resultant tensor here? In any case, there is actually more than one possible output index ordering in the sense of C_{ijk} vs C_{jki} vs C_{kij}, and so on. Thus any output index ordering is okay here.