Homework 5

Deadline: Monday, Feb. 13, at 11:59pm.

Submission: You must submit your solutions as a PDF file through MarkUs[1]. You can produce the file however you like (e.g. LaTeX, Microsoft Word, scanner), as long as it is readable.

Late Submission: MarkUs will remain open until 2 days after the deadline; until that time, you should submit through MarkUs. If you want to submit the assignment more than 2 days late, please e-mail it to `csc321ta@cs.toronto.edu`. The reason for this is that MarkUs won't let us collect the homeworks until the late period has ended, and we want to be able to return them to you in a timely manner.

Weekly homeworks are individual work. See the Course Information handout[2] for detailed policies.

1. **Regularized linear regression.** Consider a univariate linear regression problem with no bias term. This means the model predictions are given by

$$y = wx.$$

As usual, we'll use squared error loss, so the cost is given by:

$$\mathcal{E} = \frac{1}{2N}\sum_i (y^{(i)} - t^{(i)})^2 = \frac{1}{2N}\sum_i (wx^{(i)} - t^{(i)})^2,$$

where N is the number of training examples. In your calculations and/or answer, you may need to write $(\mathbf{x}^{(i)})^2$ for the ith input squared.

(a) **[1pts]** Find an expression for the weight w which minimizes the L_2-regularized loss

$$\mathcal{E}_{L_2} = \mathcal{E} + \frac{\lambda}{2}w^2.$$

(b) **[2pts]** Find an expression for the weight w which minimizes the L_1-regularized loss

$$\mathcal{E}_{L_1} = \mathcal{E} + \lambda|w|.$$

Note that the minimum will occur either at a point where $\mathrm{d}\mathcal{E}_{L_1}/\mathrm{d}w = 0$ or at a point where $\mathcal{E}_{L_1}$ is nondifferentiable. You need to consider both cases. Your answer will have the form

$$w = \begin{cases} \cdots & \text{(some condition involving the } \mathbf{x}^{(i)}\text{'s, } t^{(i)}\text{'s, and } \lambda) \\ \cdots & \text{(some condition involving the } \mathbf{x}^{(i)}\text{'s, } t^{(i)}\text{'s, and } \lambda) \\ \cdots & \text{(some condition involving the } \mathbf{x}^{(i)}\text{'s, } t^{(i)}\text{'s, and } \lambda) \end{cases}$$

2. **Dropout.** **[4pts]** Dropout has an interesting interpretation in the case of linear regression. Recall that the predictions are made stochastically as:

$$y = \sum_j m_j w_j x_j,$$

[1] `https://markus.teach.cs.toronto.edu/csc321-2017-01`

[2] `http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/syllabus.pdf`

where the m_j's are all i.i.d. (independnet and identically distributed) Bernoulli random variables with expectation 1/2. (I.e., they are indepdendent for every input dimension and every data point.) We would like to minimize the cost

$$\mathcal{E} = \frac{1}{2N} \sum_{i=1}^{N} \mathbb{E}[(y^{(i)} - t^{(i)})^2], \tag{1}$$

where the expectation is with respect to the $m_j^{(i)}$'s.

Now we show that this is equivalent to a regularized linear regression problem:

(a) Find expressions for $\mathbb{E}[y]$ and $\mathrm{Var}[y]$ for a given $\mathbf{x}$ and $\mathbf{w}$.

(b) Determine $\tilde{w}_j$ as a function of w_j such that

$$\mathbb{E}[y] = \tilde{y} = \sum_j \tilde{w}_j x_j.$$

Here, $\tilde{y}$ can be thought of as (deterministic) predictions made by a different model.

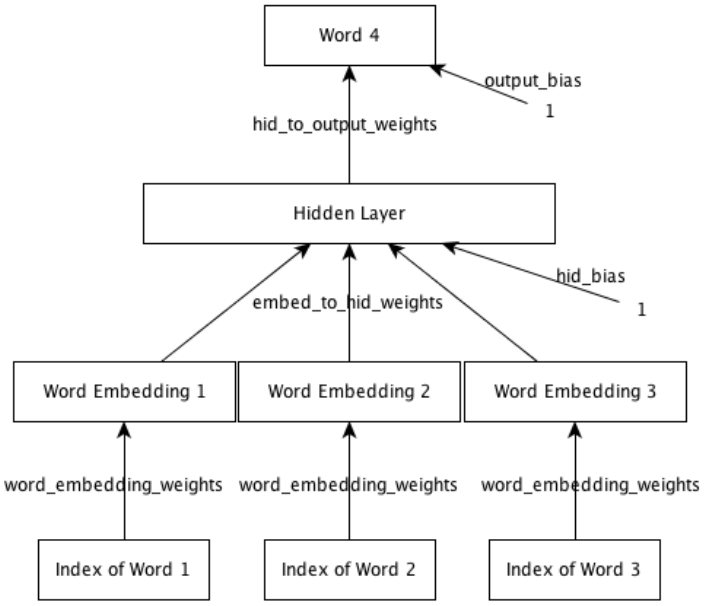
(c) Using the model from the previous section, show that the cost $\mathcal{E}$ (Eqn. 1) can be written as

$$\mathcal{E} = \frac{1}{2N} \sum_{i=1}^{N} (\tilde{y}^{(i)} - t^{(i)})^2 \ + \ \mathcal{R}(\tilde{w}_1, \ldots, \tilde{w}_D),$$

where $\mathcal{R}$ is a function of the $\tilde{w}_D$'s which does not involve an expectation. (Note that $\mathcal{R}$ will depend on the data, so we call it a "data-dependent regularizer.")

Hint: write the cost in terms of the mean and variance formulas from part (a). For inspiration, you may wish to refer to the derivation of the bias/variance decomposition from Lecture 9.

3. **Neural language model. [3pts]** In this part, we count the number of trainable parameters in a neural language model. It receives as input 3 consecutive words, and its aim is to predict a distribution over the next word (the *target* word). The model architecture is as follows:



The network consists of an input layer, embedding layer, hidden layer and output layer. The input consists of a sequence of 3 consecutive words, given as integer valued indices. (*I.e.*, the 250 words in our dictionary are arbitrarily assigned integer values from 0 to 249.) The embedding layer maps each word to its corresponding vector representation. This layer has $3 \times D$ units, where D is the embedding dimension, and it essentially functions as a lookup table. We share the *same* lookup table between all 3 positions, *i.e.* we don't learn a separate word embedding for each context position. The embedding layer is connected to the hidden layer, which uses a logistic nonlinearity. The hidden layer in turn is connected to the output layer. The output layer is a softmax over the 250 words.

As above, assume we have 250 words in the dictionary and use the previous 3 words as inputs. Suppose we use a 16-dimensional word embedding and a hidden layer with 128 units. The trainable parameters of the model consist of 3 weight matrices and 2 bias vectors.

(a) What is the total number of trainable parameters in the model? Which layer has the largest number of trainable parameters?

(b) How many add-multiply operations are needed to make predictions, assuming the first layer is implemented using a lookup table?