6.790 Homework 4

Revision: 10/21/24 11:00PM

Questions 1 and 2 are optional. The remaining problems are required. All the questions are written (don't include running code) and go over online learning, inner workings of neural network architectures, and robustness. There are some hints in the blue boxes that are supposed to help you in your solution, but you can choose to disregard them.

Please hand in your work via Gradescope via the link at https://gradml.mit.edu/info/homeworks/. If you were not added to the course automatically, please use Entry Code R7RGGX to add yourself to Gradescope. Make sure to assign the problems to the corresponding pages in your solution when submitting via Gradescope.

1. Latex is not required, but if you are hand-writing your solutions, please write clearly and carefully. You should include enough work to show how you derived your answers, but you don't have to give careful proofs.

2. Homework is due on Tuesday November 5 at 11PM.

3. Lateness and extension policies are described at https://gradml.mit.edu/info/class_policy/.

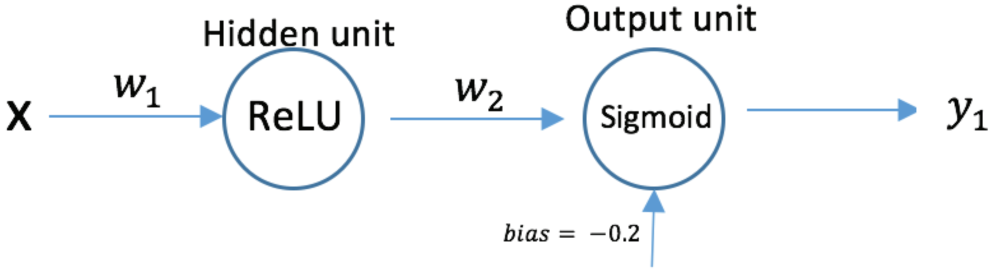
Contents

1 ReLU Backpropagation [optional]

1.1 Single output network

1. The rectified linear unit (ReLU) is a popular activation function for hidden layers. The activation function is a ramp function $f(z) = \max(0, z)$ where $z = wx$. This has the effect of simply thresholding its input at zero. Unlike the sigmoid, it does not saturate near 1 and is also simpler in gradient computations, resulting in faster convergence of SGD. Furthermore, ReLUs can allow networks to find sparse representations, due to their thresholding characteristic, whereas sigmoids will always generate non-zero values. However, ReLUs can have zero gradient when the activation is negative, blocking the backpropagation of gradients.

   Here you use a very small neural network: it has one input unit, taking in a value x, one hidden unit (ReLU), and one output unit (sigmoid). We include a bias term of +0.2 on the sigmoid unit (the figure mistakenly shows a -0.2 bias).



   We use the following quantities in this problem:

$$z_1 = w_1 x$$
$$a_1 = \mathrm{ReLU}(z_1)$$
$$z_2 = w_2 a_1 + 0.2$$
$$y = \sigma(z_2)$$

   The weights are initially $w_1 = \frac{1}{10}$ and $w_2 = -1$.

   Let's consider one training example. For that training case, the input value is $x = 2$ (as shown in the diagram), and the target output value $t = 1$. We're using the following loss function:

$$E = \frac{1}{2}(y - t)^2$$

   Please supply numeric answers; the numbers in this question have been constructed in such a way that you don't need a calculator. Show your work in case of mis-calculation in earlier steps.

   (a) What is the output of the hidden unit for this input?

(b) What is the output of the output unit for this input?

(c) What is the loss, for this training example?

(d) Write out an abstract symbolic expression for derivative of the loss with respect to w_1 as repeated applications of the chain rule. For example, for the derivative of the loss with respect to w_2, we would write $\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial w_2}$.

(e) Write the expression for each partial derivative in the chain rule expansion from the previous part. For example, $\frac{\partial y}{\partial z_2} = y(1-y)$.

(f) What is the derivative of the loss with respect to w_1, for this training example?

(g) What would the update rule for w_1 be if the learning rate is η?

(h) If η is large enough, w_1 will update from its current value of 0.1 to a negative value. Assume our new value is $w_1 = -0.1$. What will be the output of the output unit for an input of $x = 2$?

(i) What will happen when we try to update the weight, using this new example of $x = 2$, for w_1 for any value of target? Why?

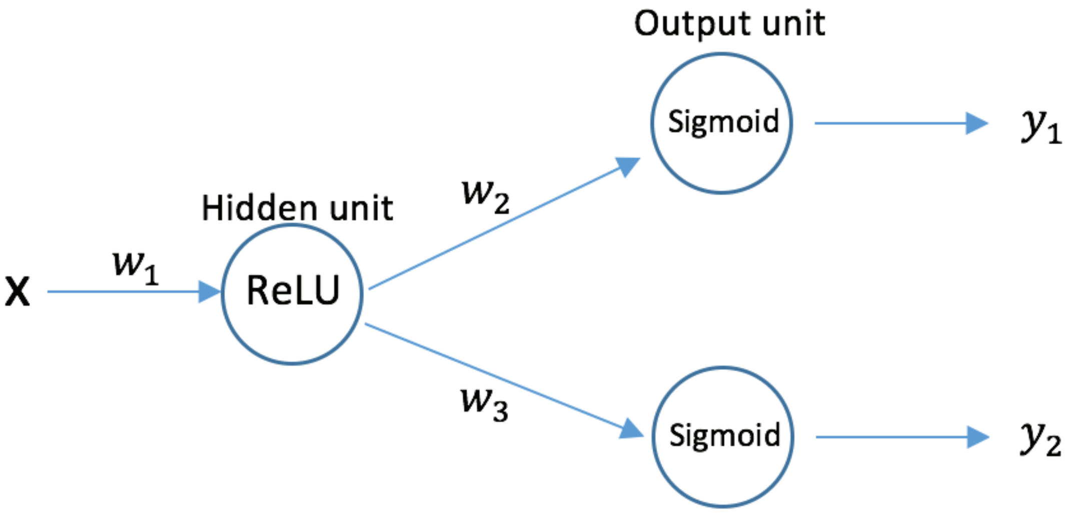
(j) Is it a bad idea to have a ReLU activation on your output layer?

(k) Consider the case of (i) where we replace our ReLU with the following activation function:

$$f(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha z & \text{if } otherwise. \end{cases}$$

for some small alpha, e.g. $\alpha = 0.01$, and $z = wx$. Does this address the problem we were facing in (i)? (this is known as dying ReLU)

1.2 Multiple output network



(a)

$$a_1 = \mathrm{ReLU}(0, w_1 x)$$

$$y_1 = \sigma(w_2 a_1)$$

$$y_2 = \sigma(w_3 a_1)$$

Write out an abstract symbolic expression for the derivative of the loss with respect to w_1 for the network above with two output units, as repeated applications of the chain rule.

Multi-output (multi-class) networks are used in many settings such as object recognition, where we are trying to classify an image as being one of K objects. Each of the K possible objects would correspond to an output unit in the network. For this purpose, the sigmoid activation and squared loss are replaced by softmax activation and cross-entropy loss. This is similar to the multi-class logistic regression we saw in the week 4 exercises.

The softmax is given by:

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \quad .$$

(b) When $K > 3$, why might sigmoid units be a bad idea?

2 Noisy Targets [10 points]

2. Consider a binary classification problem in which the true target values are $y \in \{0, 1\}$ with a network output $h(x, w)$ that represents $p(y = 1 \mid x)$. However our training set is not true-label pairs (x, y) but instead noisily labelled pairs (x, y'), where y' are the labels y that have each been independently flipped with probability ϵ.

   (a) Assuming independent and identically distributed data, write down the error function corresponding to the negative log likelihood. Verify that the *cross entropy* error function is obtained when $\epsilon = 0$. Note that this error function makes the model robust to incorrectly labeled data, in contrast to the usual error function.

   (b) What is the form of the partial derivative with respect to a single weight in the output layer?

   (c) How does the stochastic gradient update rule for $\epsilon = 0.1$ differ from the case when $\epsilon = 0$?

3 Architecture Details [10 pts]

3.1 Activations

3. (a) Consider a neural network in which layer $\ell - 1$ takes in some pre-activations $a^{(\ell-1)}$, applies the ReLU activation to get the activations of $z^{(\ell-1)}$ of layer $\ell - 1$, and then applies a randomly initialized linear layer ℓ to compute the pre-activations $a^{(\ell)}$ of layer ℓ:

$$a_i^{(\ell)} = \sum_{j=1}^{M} w_{ij} z_j^{(\ell-1)}, \qquad z_i^{(\ell-1)} = \text{ReLU}(a_i^{(\ell-1)})$$

   Suppose we initialize the weights $w \sim N(0, \epsilon^2)$ and the pre-activations of the previous layer $a^{(\ell-1)} \sim N(0, \lambda^2)$. Find the setting of ϵ that keeps the pre-activations of the next layer $a^{(\ell)}$ distributed the same way $a^{(\ell)} \sim N(0, \lambda^2)$.

3.2 Convolutional Neural Networks

(a) Consider a convolutional neural network layer that takes in a 1d input array of length 5 and applies a feature map that is a convolutional filter of width 3 with stride 1. Show that this layer is a special case of a fully connected MLP layer by writing out the matrix of weights that this layer acts like, using shared variables for shared weights and putting 0 for nonexistent connections. Ignore any bias terms.

3.3 Transformers

(a) Express the self-attention function given by

$$Y = \text{Softmax}(\frac{QK^T}{\sqrt{d}})V$$

as a fully connected network in the form of a matrix that maps the full input sequence of concatenated word vectors into an output vector of the same dimension. Note that such a matrix would have $O(N^2d^2)$ parameters. Show that the self-attention network corresponds to a sparse version of this matrix with parameter sharing. Draw a sketch showing the structure of this matrix, indicating which blocks of parameters are shared and which blocks have all elements equal to zero.

(b) Show that if we omit the positional encoding of input vectors then the outputs of a multi-head attention layer are equivariant with respect to a reordering of the input sequence.

(c) Consider the positional encoding scheme where for position n the elements r_{ni} of the 2d dimensional positional encoding r_n are given for $i = 0, 1, \cdots, 2d - 1$ by:

$$r_{ni} = \begin{cases} \sin\left(\frac{n}{L^{i/D}}\right), & \text{if } i \text{ is even} \\ \cos\left(\frac{n}{L^{(i-1)/D}}\right), & \text{if } i \text{ is odd} \end{cases}$$

Show that this scheme has the property that for any fixed integer k, there is a $2d \times 2d$ matrix W_k, only a function of k, such that $r_{n+k} = W_k r_n$ for all integer n.

Show that if the encoding is based purely on sine functions, without cosine functions, then this property no longer holds.

Hint: Make use of the following trigonometric identities:

$$\cos(A + B) = \cos(A)\cos(B) - \sin(A)\sin(B), \qquad \sin(A + B) = \cos(A)\sin(B) + \sin(A)\cos(B)$$

3.4 Graph Neural Networks

(a) Show that a graph attention network in which the graph is fully connected, so that there is an edge between every pair of nodes, is equivalent to a standard transformer architecture.

4 Online Logistic Regression [10 points]

4. Consider logistic regression for a data set $\{(x^{(i)}, y^{(i)})\}$ with $y^{(i)} \in \{-1, 1\}$. The loss function for each sample $(x^{(i)}, y^{(i)})$ with weight vector w is given by

$$\ell_i(w) = \log(1 + \exp\{-y^{(i)} w^\top x^{(i)}\}) + \lambda \|w\|^2 . \tag{5}$$

Assume $\lambda = 1$ for this problem.

(a) Derive the online convex optimization (OCO) algorithm with online gradient descent scheme for a sequence of functions $\{\ell_i\}$ with step-size η. In other words, if $w^{(i)}$ is the weight vector used to predict the i-th example. Write down equation for $w^{(i+1)}$ in terms of the $w^{(i)}, x^{(i)}, y^{(i)}$, and η. Assume that there is no restriction on the domain set of each $w^{(i)}$, i.e. if $x^{(i)} \in \mathbb{R}^p$, then $w^{(i)} \in B = \mathbb{R}^p$.

(b) (optional) We know from the lecture that cross-entropy loss used in online logistic regression is convex. From this, which of the following is true? Give a short explanation.

- $\dfrac{\ell_i(w^*) - \ell_i(w^{(i)})}{\|w^* - w^{(i)}\|} \geqslant \nabla \ell_i(w^{(i)}) \cdot \dfrac{w^* - w^{(i)}}{\|w^* - w^{(i)}\|}$
- $\dfrac{\ell_i(w^*) - \ell_i(w^{(i)})}{\|w^* - w^{(i)}\|} \leqslant \nabla \ell_i(w^{(i)}) \cdot \dfrac{w^* - w^{(i)}}{\|w^* - w^{(i)}\|}$

Here, the right hand side represents the slope of ℓ_i at $w^{(i)}$ in the direction that point from $w^{(i)}$ to w^*.

(c) (optional) Use rule of cosine as well as the online gradient descent scheme, show that

$$\|w^{(i+1)} - w^*\|^2 - \|w^{(i)} - w^*\|^2 = \eta^2 \|\nabla \ell_i(w^{(i)})\|^2 - 2\eta \nabla \ell_i(w^{(i)}) \cdot (w^{(i)} - w^*). \tag{6}$$

(d) (optional) In class, we mentioned an upper bound on the quantity $L_n - L_*$ (known as regret) where

$$w_* = \arg\min_w \sum_{i=1}^n \ell_i(w), \tag{7}$$

$$L_* = \frac{1}{n} \sum_{i=1}^n \ell_i(w^*), \tag{8}$$

and

$$L_n = \frac{1}{n} \sum_{i=1}^n \ell_i(w^{(i)}). \tag{9}$$

Use these definitions , and results from previous two sub-problems to show that

$$L_n - L_* \leqslant \frac{1}{2n} \left(\frac{1}{\eta} \|w^{(1)} - w^*\|^2 + \eta \sum_{i=1}^n \|\nabla \ell_i(w^{(i)})\|^2 \right) \tag{10}$$

(e) Suppose A priori you only know very coarse information about the data set: you know n, $w^{(1)} = 0$, $\|w^*\| \leqslant 1$, and that $\|x^{(i)}\| \leqslant 77$. What value of η would you choose for running OCO? (Hint: try upper-bounding $\|\nabla \ell_i(w^{(i)})\|$.)

5 Adversarial Example [10 points]

5. Willy Makeit has trained up a one-layer neural network with a sigmoid activation function to classify ferns based on several important features of their leaves. The hypothesis is:

$$h(x; W, W_0) = \sigma(W^T x + W_0) \ .$$

Willy is particularly excited to find that this network correctly classifies an important but unusual-looking species of fern (which we will call x^*) as a positive example.

We expect ferns with extremely similar features to x^* to be of the same class as x^* so we'd expect them to also always be classified positively if Willy's classifier is good. Betty Wont wants to defeat Willy's classifier by finding another fern, x_A, that is very similar to x^* but which Willy's classifier predicts is negative.

The problem Betty wants to solve can be framed as finding a new input $x_A = \arg\min_x J(x)$, where

$$J(x) = \alpha\|x - x^*\|^2 + \max(0, h(x; W, W_0) - 0.5) \ .$$

(a) Which term in the objective J depends on the class of x? Explain in words what it is computing and why it makes sense in this problem.

(b) Betty thinks gradient descent would be a good way to solve this problem. If $x \in \mathbb{R}^d$, what are the dimensions of $\nabla_x J(x)$?

(c) Write an expression for

$$\nabla_x J(x)$$

in terms of W, W_0, and x. Recall that $\sigma(z) = \frac{e^z}{e^z+1}$ and $\sigma'(z) = \sigma(z)(1 - \sigma(z))$.

(d) If Betty sets α to a very *small* value and finds $x_A = \arg\min_x J(x)$, is it likely that she will have succeeded in finding a plant similar to x^* that is classified as negative? Explain why or why not.

(e) If Betty sets α to a very *large* value and finds $x_A = \arg\min_x J(x)$, is it likely that she will have succeeded in finding a plant similar to x^* that is classified as negative? Explain why or why not.

Hint: it might be helpful (but not strictly necessary) to look at the logistic distribution in order to avoid complicated integral calculations.