6.790 Homework 4

Revision: 10/21/24 11:00PM **Solutions**

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

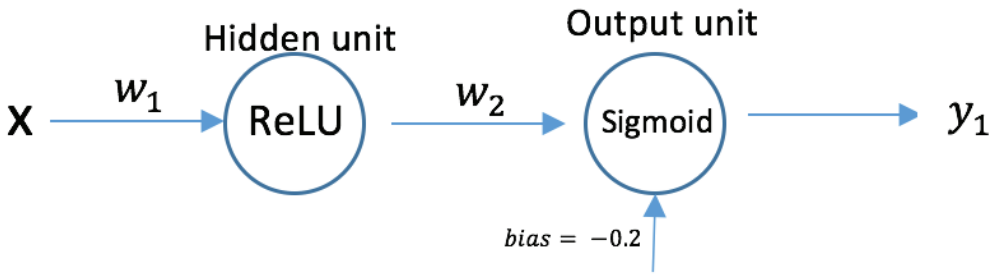
Solution: Don't look at the solutions until you have tried your absolute hardest to solve the problems. This is especially true for optional problems that you didn't work on—it's a good idea to come back to them when studying for exams.

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:

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

   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?

Solution:

$$a = \text{ReLU}(z_1) = \max(0, w_1 x) = \max(0, \frac{1}{10} \times 2) = \frac{1}{5}$$

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

Solution:

$$y = \sigma(w_2 \max(0, w_1 x)) = \sigma(-1 \times \frac{1}{5} + 0.2) = \sigma(0) = \frac{1}{2}$$

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

Solution:

$$E = \frac{1}{2}(y - t)^2 = \frac{1}{2}(\frac{1}{2} - 1)^2 = 1/8$$

(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 $\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial w_2}$.

Solution: $\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial a}\frac{\partial a}{\partial z_1}\frac{\partial z_1}{\partial w_1}$

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

Solution: $\frac{\partial E}{\partial y} = y - t$

$\frac{\partial y}{\partial z_2} = y(1-y)$

$\frac{\partial z_2}{\partial a} = w_2$

$\frac{\partial a}{\partial z_1} = \begin{cases} 1 & \text{if } w_1 x > 0 \\ 0 & \text{if } w_1 x < 0. \end{cases} = I[w_1 x > 0]$ (indicator function)

$\frac{\partial z_1}{\partial w_1} = x$

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

Solution:

$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial a}\frac{\partial a}{\partial z_1}\frac{\partial z_1}{\partial w_1} =$

$(y - t) \times y(1-y) \times w_2 \times I[w_1 x > 0] \times x =$

$(\frac{2}{5} - 1) \times \frac{2}{5}\frac{3}{5} \times -1 \times 1 \times 2 = 36/125 = .288$

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

Solution:

$$w_1 := w_1 - \eta \frac{\partial E}{\partial w_1} := w_1 + .288\eta$$

(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 $x = 2$?

Solution: The ReLU will output 0, since w_1 is negative, so only the bias term will remain in the sigmoid and the output will be $\sigma(-0.2)$

(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?

Solution: The ReLU gate is closed and gradients will not flow backwards through the ReLU unit. w_1 will not be updated with SGD. In fact, if the training examples are all positive, the input to the ReLU will be always be negative, effectively killing the ReLU.

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

Solution: Yes, if the input to the ReLU is mostly negative, it will fail to backpropagate gradients through the entire network.

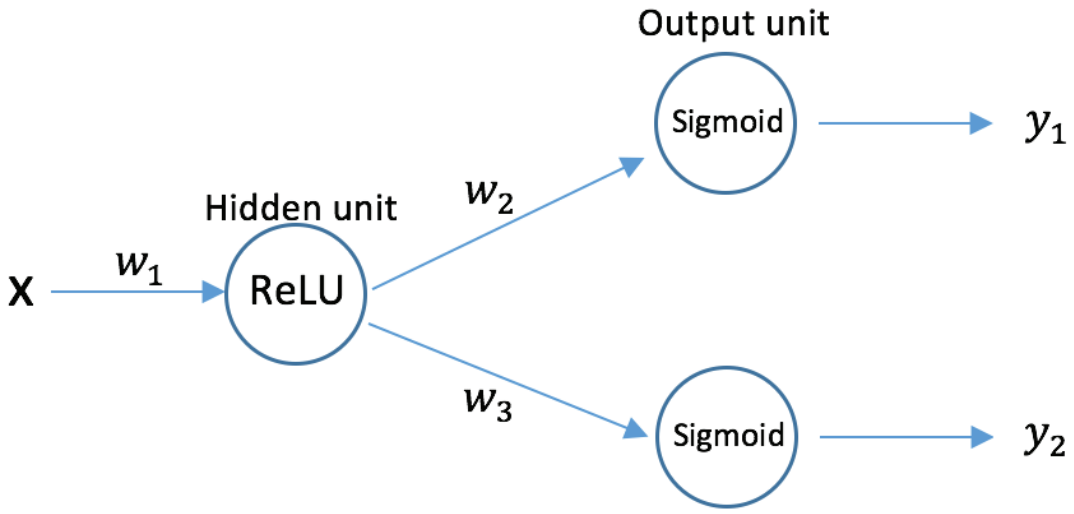
(k) Consider the case of (i) where we replace our ReLU with the following activation function:

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

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)

Solution: ReLU units don't backpropagate any error for negative inputs. The leaky ReLU allows a small error to backpropagate even with negative input.

1.2 Multiple output network



(a)

$$a_1 = \text{ReLU}(0, w_1 x)$$
$$y_1 = \sigma(w_2 a_1)$$
$$y_2 = \sigma(w_3 a_1)$$

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.

Solution:

$$E_{total} = \frac{1}{2}(y_1 - t_1)^2 + \frac{1}{2}(y_2 - t_2)^2 = E_1 + E_2$$

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_1}{\partial w_1} + \frac{\partial E_2}{\partial w_1}$$

$$\frac{\partial E_1}{\partial w_1} = \frac{\partial E_1}{\partial y} \frac{\partial y}{\partial a_1} \frac{\partial a_1}{\partial w_1}$$

Similarly, for E_2

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?

Solution: With sigmoid output units for multiple classes, we cannot guarantee that at most one output unit activates. The normalization in the softmax makes this happen

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.

   Solution: Let $t^{(n)} \in \{0, 1\}$ denote the dataset label and $y^{(n)} \in \{0, 1\}$ denote the true class label for $x^{(n)}$. From the rules of probability we have

   $$p(t = 1|x) = \sum_{y=0}^{1} p(t = 1|y)p(y|x) = (1 - \epsilon)h(x^{(n)}, w) + \epsilon(1 - h(x^{(n)}, w)) \tag{1}$$

   The conditional probability of the data label is then

   $$p(t|x) = p(t = 1|x)^t(1 - p(t = 1|x))^{1-t} \tag{2}$$

   Forming the likelihood and taking the negative logarithm we have the error function in the form

   $$E(w) = -\sum_{n=1}^{N} \Big\{ t^{(n)} \log \Big[(1 - \epsilon)h(x^{(n)}, w) + \epsilon(1 - h(x^{(n)}, w)) \Big] + \tag{3}$$

   $$(1 - t^{(n)}) \log \Big[1 - (1 - \epsilon)h(x^{(n)}, w) - \epsilon(1 - h(x^{(n)}, w)) \Big] \Big\} \tag{4}$$

   (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:

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

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

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

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 $O(N^2 d^2)$ 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 $w^{(i)} \in B = \mathbb{R}^p$.

Solution: With online gradient descent, the update step is $w^{(i+1)} = w^{(i)} - \eta \nabla \ell_i \left(w^{(i)}\right)$ for each sample $\left(x^{(i)}, y^{(i)}\right)$.

We know $\nabla \ell_i(w) = \dfrac{-y^{(i)} x^{(i)} \exp\{-y^{(i)} w^\top x^{(i)}\}}{1 + \exp\{-y^{(i)} w^\top x^{(i)}\}} + 2w = \dfrac{-y^{(i)} x^{(i)}}{1 + \exp\{y^{(i)} w^\top x^{(i)}\}} + 2w.$

Hence, $w^{(i+1)} = w^{(i)} - \eta \left(\dfrac{-y^{(i)} x^{(i)}}{1 + \exp\{y^{(i)} w^{(i)\top} x^{(i)}\}} + 2w^{(i)} \right)$

(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^*.

Solution: Since the function is convex, any point between $w^{(i)}$ and w^* must be below the straight line connecting $(w^{(i)}, \ell_i(w^{(i)}))$, and $(w^*, \ell_i(w^*))$. This means that the tangent line at $w^{(i)}$ must has lower gradient in the direction that point from $w^{(i)}$ to w^* than that line, i.e.

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

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

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

Solution: From the online gradient descent scheme,

$$w^{(i+1)} - w^* = (w^{(i)} - w^*) - \eta \nabla \ell_i(w^{(i)}).$$

Use rule of cosine,

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

(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

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

Solution: From the sub-problem 3, we can rearrange the terms, and get

$$2\eta \nabla \ell_i(w^{(i)}) \cdot (w^{(i)} - w^*) = \eta^2 \|\nabla \ell_i(w^{(i)})\|^2 - \|w^{(i+1)} - w^*\|^2 + \|w^{(i)} - w^*\|^2,$$
$$\nabla \ell_i(w^{(i)}) \cdot (w^{(i)} - w^*) = \frac{1}{2} \left(\eta \|\nabla \ell_i(w^{(i)})\|^2 + \frac{1}{\eta} \left(\|w^{(i)} - w^*\|^2 - \|w^{(i+1)} - w^*\|^2 \right) \right).$$

Substitute this into the result from sub-problem 2 gives

$$\ell_i(w^{(i)}) - \ell_i(w^*) \leqslant \frac{1}{2}\left(\eta\|\nabla\ell_i(w^{(i)})\|^2 + \frac{1}{\eta}\left(\|w^{(i)} - w^*\|^2 - \|w^{(i+1)} - w^*\|^2\right)\right).$$

Hence,

$$\begin{aligned}
L_n - L_* &= \frac{1}{n}\sum_{i=1}^{n}(\ell_i(w^{(i)}) - \ell_i(w^*)) \\
&\leqslant \frac{1}{2n}\sum_{i=1}^{n}\left(\eta\|\nabla\ell_i(w^{(i)})\|^2 + \frac{1}{\eta}\left(\|w^{(i)} - w^*\|^2 - \|w^{(i+1)} - w^*\|^2\right)\right) \\
&= \frac{1}{2n}\left(\frac{1}{\eta}\left(\|w^{(1)} - w^*\|^2 - \|w^{(n+1)} - w^*\|^2\right) + \eta\sum_{i=1}^{n}\|\nabla\ell_i(w^{(i)})\|^2\right) \\
&\leqslant \frac{1}{2n}\left(\frac{1}{\eta}\|w^{(1)} - w^*\|^2 + \eta\sum_{i=1}^{n}\|\nabla\ell_i(w^{(i)})\|^2\right).
\end{aligned}$$

(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 $\|\nabla\ell_i(w^{(i)})\|$.)

Solution: From the optimal η^* in sub-problem 5, we must first upper bound the sum of loss gradient in the denominator. In this way, if we choose η to minimize an upperbound of gradients of regret, then weâĂŹre minimizing the gradients of regret by proxy.

We notice that from sub-problem 1,

$$\left\|\nabla\ell_i(w^{(i)})\right\| = \left\|\frac{-y^{(i)}x^{(i)}}{1+\exp\{y^{(i)}w^{(i)\top}x^{(i)}\}} + 2w^{(i)}\right\|$$

By triangle inequality, we can bound this as

$$\begin{aligned}
\left\|\frac{-y^{(i)}x^{(i)}}{1+\exp\{y^{(i)}w^{(i)\top}x^{(i)}\}} + 2w\right\| &\leqslant \left\|\frac{-y^{(i)}x^{(i)}}{1+\exp\{y^{(i)}w^{(i)\top}x^{(i)}\}}\right\| + \|2w^{(i)}\| \\
&\leqslant \|x^{(i)}\| + 2\|w^{(i)}\| \\
&\leqslant 77 + 2 * 1 = 79
\end{aligned}$$

Note, $\|w^{(i)}\| \leqslant 1$ because we can apply projected gradient descent to this OCO problem. Given that we are certain that $\|w^*\| \leqslant 1$, we can say that $\|w^{(i)}\| \leqslant 1$ after the projection step for all t.

Therefore, the denominator term is

$$\left(\sum_{i=1}^{n}\|\nabla\ell_i(w^{(i)})\|^2\right)^{\frac{1}{2}} \leqslant \left(\sum_{i=1}^{n}79^2\right)^{\frac{1}{2}} = 79\sqrt{n}.$$

Hence, we should choose

$$\eta = \frac{1}{79\sqrt{n}}.$$

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.

Solution:

(a) **Solution:** The second term depends on the class of x. If the class of x is negative, Willy's classifier h outputs a value < 0.5, in which case, the second term evaluates to 0. Otherwise, for x of positive class, h outputs a value > 0.5, and the second term is this output minus 0.5, a positive value. Thus, this term will penalize positive predictions, with a greater loss the farther these predictions are from 0.5. However, once the prediction goes below 0.5, Betty is happy since the fern is now misclassified, so the loss goes to 0.

The first term has no depency on the class of x, only on the distance between x and x^*.

(b) **Solution:** $1 \times d$. Since we are taking the gradient of a scalar with respect to a d-dimensional vector x, the shape of the output should match the shape of x.

(c) **Solution:** We take the gradient of each term separately:

$$J(x) = J_1(x) + J_2(x),$$

where $J_1(x) = \alpha\|x - x^*\|^2 = \alpha(x-x^*)^T(x-x^*)$ and $J_2(x) = \max(0, h(x; W, W_0) - 0.5)$. We have

$$\nabla_x J_1(x) = 2\alpha(x - x^*).$$

If $h(x; W, W_0) \leqslant 0.5$, this term evaluates to 0 so the gradient is the 0 vector (with the shape of x). Otherwise, applying the chain rule,

$$\begin{aligned}\nabla_x(h(x; W, W_0) - 0.5) &= \sigma(W^T x + W_0)(1 - \sigma(W^T x + W_0))\nabla_x(W^T x + W_0) \\ &= \sigma(W^T x + W_0)(1 - \sigma(W^T x + W_0))W \\ &= h(x; W, W_0)(1 - h(x; W, W_0)W\end{aligned}$$

Thus, we have

$$\nabla_x J(x) = 2\alpha(x - x^*) + \begin{cases} 0 & \text{if } h(x; W, W_0) \leqslant 0.5 \\ h(x; W, W_0)(1 - h(x; W, W_0))W & \text{otherwise.} \end{cases}$$

(d) **Solution:** Yes, surprisingly. (Almost all of the staff got this one wrong).

When α is positive but very small, then Betty will find the plant that is closest to x^*, but classified as negative. This is because, for any negative example, the second term is 0. Then, within the space of x's that are negative, the first term will push the solution as close as possible to x^*.

(e) **Solution:** No. Here, the stress of the objective function shifts to having x^A be close to x^*, at the expense of having x^A get misclassified. Thus, she'll likely get something very similar to x^* that does not receive a negative classification.