CS 6923: Machine Learning

Fall 2019

Homework 8

Submit on NYU Classes by Nov 17 at 6:00 PM. You may work together with one other person on this homework. If you do that, hand in JUST ONE homework for the two of you, with both of your names on it. You may *discuss* this homework with other students but YOU MAY NOT SHARE WRITTEN ANSWERS OR CODE WITH ANYONE BUT YOUR PARTNER.

IMPORTANT SUBMISSION INSTRUCTIONS: Submit one file with the answers to the questions in Part I, another file with the answers to the written questions in Part II, and a third file with your code. Do NOT include the answers to part II questions in your code file.

1. In this exercise you will get practice with performing gradient descent in one variable.

   Consider the function

   $$f(x) = +4x^4 - 15x^3 + 11x^2 + 10x + 2$$

   Graph this function for x in the interval [-1,3]. You will see that it has two minima in that range, a local minimum and a global minimum. These are the only minima of the function.

   (a) What is the value of x at the local minimum and at the global minimum? Find the answer to this question like either by hand with calculus or using software.

   (b) Suppose we apply gradient descent to this function, starting with $x = -1$. To do this, we will need to update x using the update rule

   $x = x + -\eta * f'(x)$

   where f' is the derivative of f, and η is the "step size".

   Write a small program implementating gradient descent for this function. Setting $x = -1$ and $\eta = 0.001$, run gradient descent for 6 iterations (that is, do the update 6 times). Report the values of x and $f(x)$ at the start and after each of the first 6 iterations.

   Run the gradient descent again, starting with x=-1, for 1200 iterations. Report the last 6 values of x and f(x).

   Has the value of x converged? Has the gradient descent found a minimum? Is it the global or the local minimum?

   You do NOT have to hand in your code.

   (c) Repeat the previous exercise, but this time, start with x=3.

   (d) Setting $x = -1$ and $\eta = 0.01$, run gradient descent for 1200 iterations. As in the previous two exercises, report the initial values of x and f(x), the next 6 values of x and f(x), and the last 6 values of x and f(x). Compare the results obtained this time to the results obtained above for $x = -1$ and $\eta = 0.001$. What happened?

   (e) Setting $x = -1$ and $\eta = 0.1$, run gradient descent for 100 iterations. What happened?

2. Given a neural network with 3 layers, where the *input* layer has 2 neurons, the *hidden* layer has 2 neurons, the *output* layer has 1 neuron, and

$$W^{(1)} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix},\ W^{(2)} = \begin{pmatrix} 1 & 2 \end{pmatrix},\ b^{(1)} = \begin{pmatrix} 1 \\ -1 \end{pmatrix},\ b^{(2)} = \begin{pmatrix} 1 \end{pmatrix}$$

Suppose you had the following training set[1]: $((1,0)^T, 1), ((0,1)^T, 0)$. Perform 1 step of gradient descent where the learning rate is 0.2, and activation function is the sigmoid function.

3. In class, we considered a neural net with error function $J = 1/2(\mathbf{y} - \hat{\mathbf{y}})^2$. In this question, we will also discuss neural nets for three other types of problems.

To make it easier to refer to the four types of neural nets, we'll give them names. We'll call the neural net discussed in class NeuralNetRZeroOne (R for regression, since it is often used for regression tasks). We now give the names of the other three types of neural nets, with their descriptions. **NeuralNetRZeroOne:** This neural net is designed for problems with K outputs where each output is either an element in the set $\{0, 1\}$, or a real value in the interval $[0, 1]$. This neural net has sigmoid activation functions in both the hidden nodes AND the output nodes.

NeuralNetRK: The error function is squared error, the same as for NeuralNetRZeroOne. This neural net has sigmoid activation functions in both the hidden nodes, but NOT the output nodes. Each output node just outputs the z score for that node directly.This is equivalent to saying that the activation function at the output nodes is the identity function.

NeuralNetCB: NeuralNetCB is a standard neural net for binary classification, It has a single output node which outputs a value $\hat{y}$, where $\hat{y}$ is the predicted value of $P[Class1|x]$. It uses sigmoid functions as the activation functions both for the hidden nodes and for the output nodes. [2] The goal of the backpropagation is to minimize cross-entropy error. (i.e. The loss function is changed to use the cross-entropy error instead of the squared error,.)

NeuralNetCK: This neural net is for classification with $K > 2$ classes. There are K output nodes i, and y_i is the predicted value of $P[Classi|\mathbf{x}]$. In this case, each output $\hat{y}_i = a_i^{(n_\ell)}$ is equal to $\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ (Where z_j is the z score for the j'th neuron of the output layer.) Note that the sum of the $\hat{y}_i$ is 1, which is appropriate since the y_i are the estimated probabilities for the K labels of $\mathbf{x}$. (In fact, the denominator in the expression for $\hat{y}_i$ is just a "normalizing factor" that causes the sum of the $\hat{y}_i$ to equal 1.) The error function is the generalization of cross-entropy to K classes: $-\sum_{i=1}^{K} y_i \log \hat{y}_i$, where $y_i = 1$ if $\mathbf{x}$ is in Class i, and $y_i = 0$ otherwise.

(a) Which of the above neural nets can output values that are negative numbers?

(b) Which of the above neural nets ensures that outputs $y_1, \ldots, y_K$ will satisfy $\sum_{i=1}^{K} = 1$?

(c) Consider a simple image classification problem. Each image is a 10x10 array of pixels, and all pixel values are between 0 and 1. These pixel arrays are treated as example vectors of length 100, with one attribute per pixel. There are three categories: face, cat, and tree. These are represented by 3 output values, y_1, y_2, and y_3 where y_1 is 1 or 0 depending on whether or not the image is a face, y_2 is 1 or 0 depending on whether it is a cat, and y_3 is 1 or 0 depending on whether it is a tree.

Suppose you want to use a neural net that will take an input image, and output three values p_1, p_2, p_3, where p_1 is the probability the image is a face, p_2 that it is a cat, and p_3 that it is a tree.

[1] We are representing the training set using: $((x_1, x_2)^T, y)$

[2] Another popular choice of activation function for such a network is *tanh*, the hyperbolic tangent function. Deep nets often use ReLU (Rectified Linear Units) in the hidden nodes.

Which of the above neural nets would be appropriate for the image classification problem and why?

(d) Consider the following text classification problem. Each example is a document, represented by a binary vector of length n. Each attribute corresponds to a word, and the value is 1 or 0 depending on whether the word appears in the document. There are two outputs. The first is 1 or 0 depending on whether or not the document is about politics. The second is 1 or 0 depending on whether it is written in a formal or informal style.

Which of the above neural nets would be appropriate for the text classification problem and why?

4. Consider a classification problem where examples correspond to mushrooms, and the task is to determine whether the mushroom is edible (1) or poisonous (0) [3] Suppose one of the attributes is "odor" and it has the following 4 values:

- almond
- anise
- creosote
- fishy

There are actually two types of categorical variables (attributes): nominal and ordinal. The odor attribute is a nominal attribute, meaning the values (almond, anise, creosote, fishy) are not ordered.

The values of ordinal variables are ordered: e.g. if the values are low, medium, high, we would say that low $<$ medium $<$ high.

(Note: Sometimes in Machine Learning people talk about "nominal attributes" when they really mean "categorical attributes".)

(a) In order to use a neural net on this dataset, we need to decide how to convert the categorical values to numerical values. The obvious way to do this is to just assign a number to each value: almond (1), anise (2), creosote (3), fishy (4). This approach (directly converting attribute values to numbers) is sometimes called "label encoding".

This is NOT a good way to convert nominal values to numerical values, for use in a neural net. However, it would be fine to do this if we were using a random forest, rather than a neural net. Why?

(b) When converting nominal values to numerical values for neural nets (and a number of other learning methods), the following method is often used instead. It is often called one-hot encoding. In this method, for each possible value v of an attribute, we create a new binary-valued attribute whose value is 1 if the original attribute equals v, and 0 otherwise. We use these new attributes in place of the original attribute.

For example, we would replace the odor attribute above by 4 attributes we'll call z_1, z_2, z_3, z_4, where $z_1 = 1$ iff odor=almond, $z_2 = 1$ iff odor = anise, $z_3 = 1$ iff odor=creososte, and $z_4 = 1$ iff odor = fishy. If odor was the only attribute, and the original dataset was as follows

	odor	label
$x^{(1)}$	anise	0
$x^{(2)}$	creososte	1

then using one-hot encoding to convert the input attribute, the transformed dataset would be as follows:

[3] This question is inspired by the well-known mushroom dataset, https://archive.ics.uci.edu/ml/datasets/mushroom which was used in many Machine Learning experiments.

	z_1	z_2	z_3	z_4	label
$x^{(1)}$	0	1	0	0	0
$x^{(2)}$	0	0	1	0	1

i. Now suppose that the original dataset actually has two attributes, the odor attribute just described, and another attribute called *stalk shape*. The stalk-shape attribute has two possible values, tapering or enlarging.
What is the transformed dataset, if you apply one-hot encoding to the attributes in the following dataset?

	odor	stalk shape	label
$x^{(1)}$	fishy	tapering	0
$x^{(2)}$	creososte	enlarging	0

ii. Consider an ordinal attribute whose values are low, medium, and high. It might be better to represent these values as 1,2, and 3 rather than using one-hot-encoding. Why?

iii. The stalk shape attribute has only two values. For nominal attributes with only two values, it's generally fine to just represent the two values as 0 and 1 (or -1 and +1), rather than using one-hot encoding as described above. Why is this the case for attributes with 2 values, but not for attributes with more than two values?
(Notes: If the label values are categorical, we also need to convert them to numerical values for use in a neural net. Above, we converted "poisonous" and "edible" into 1 and 0.)

5. What is the result of applying the 3×3 horizontal edge detection filter (with $S = 1$) on the following matrix:

192	171	137	81
165	123	79	42
126	72	27	15
70	35	14	10

6. Add padding to the matrix in question 5 so that after the convolution the activation map is the same as the matrix (same convolution).

7. What is the size of the activation map if you apply a 2×2 filter with pad $P = 1$, and a stride $S = 2$ to the matrix in question 5?

8. (Do not turn in) Apply a 2×2 max pool above with stride 2 to the matrix in question 5.

9. (Do not turn in) If you applied a 3×3 max pool with stride 1 to the matrix in question 5, what is the size of the activation map?

10. (Do not turn in) Suppose we had a small convolutional neural network that converted a 12×12 image into 3 output values. Where the 12×12 input connected to a convolution layer with 3 filters where the filters were of size 3×3 and a stride of one. Next a sigmoid activation was performed on the output of the convolutional layer. Next a max pool layer with filter of size 2×2 was applied with a stride of 2. The output of max pool layer was flatten and a fully connected layer was sent to the final layer with a softmax activation function.

- How many weights in the convolutional layer need to be learned?
- How many weights are learned for the entire network?
- How many sigmoid operations are performed when predicting a value?

Part II: Programming Exercise

11. Modify the neural network implementation we discussed in class to see if you can improve the performance on the MINST dataset by trying the following:

    (a) Add a regularization term to the cost function $\frac{\partial J(W,b)}{\partial W_{ij}^{(\ell)}} = \frac{1}{n}\left[\sum_{i=1}^{n} \frac{\partial J(W,b,\mathbf{x}^{(i)},y^{(i)})}{\partial W_{ij}^{(\ell)}}\right] + \frac{\lambda}{2} W_{ij}^{(\ell)}$ where $\mathbf{x}^{(i)}, y^{(i)}$ are the ith training example. See section 1.2 in `http://adventuresinmachinelearning.com/improve-neural-networks-part-1/`

    (b) Try using the *ReLU* activation function, $f(z) = \max(0, z)$. You will notice it is not differentiable at 0, but you can use: $f'(z) = 0$ if $z < 0$ and $f'(z) = 1$ if $z \geq 0$. (You can also try using the *leaky ReLU* activation function.) For more information see `https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning`

    (c) Try using the *tanh* activation function, $f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$. The derivative of *tanh* is $f'(z) = 1 - (f(z))^2$. For more information see `http://ufldl.stanford.edu/wiki/index.php/Neural_Networks`

    (d) Try the different weight initializations given in the lecture notes

    (e) Experiment on your own trying different hyper-parameters

    (f) Report your findings in a chart. The GA's will post the form of the chart you should use by Friday, Nov 8th