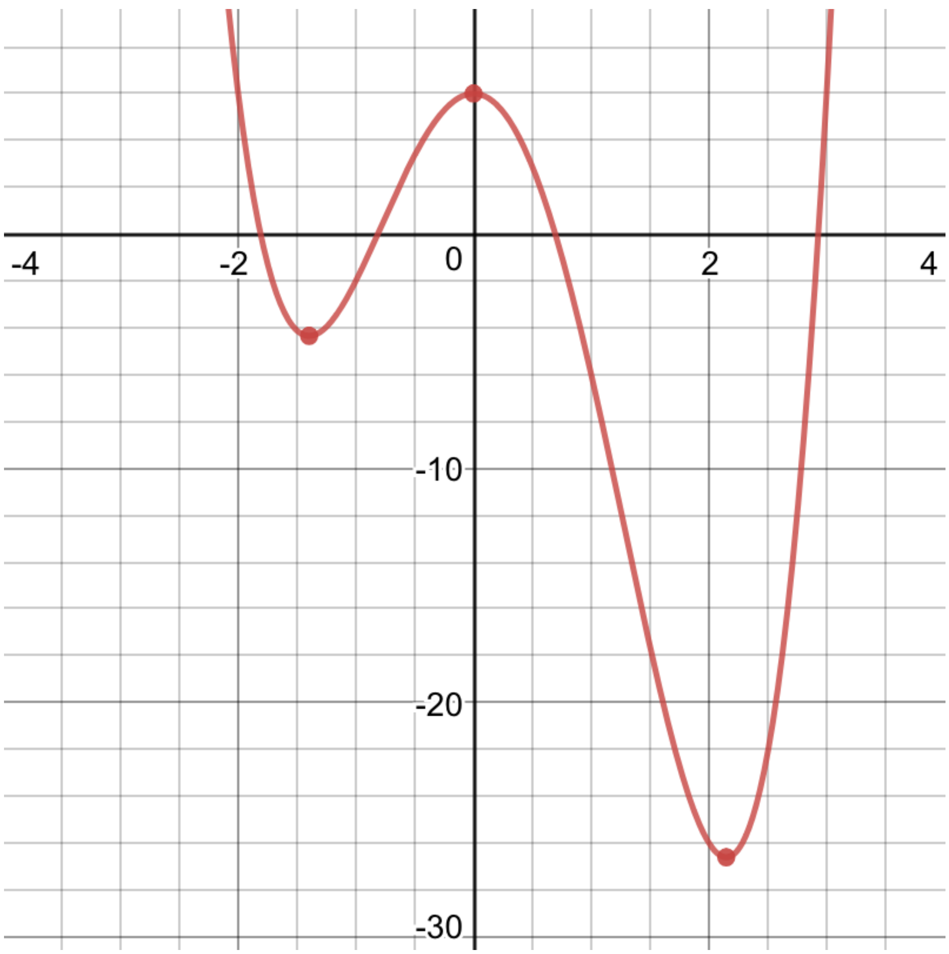
Homework 04 — ML Fall 2018

amc1354 & ads798

November 2018

1 Problem1



$2x^4 - 2x^3 - 12x^2 + 6$

(a) $x_{local\ min} = -1.3971808598447282$
$x_{global\ min} = +2.1471808598447284$.

(b) For this and the following sub-problems, in our implementation of gradient descent, we used a precision parameter of 0.000001 to give a solution close enough to the x coordinate of the real minimum.

Starting from $x = -4$, using 6 iterations and a learning rate $\eta = 0.001$, we have:

Iteration 0:
$x = -4$
$f(x) = 454$
Iteration 1:
$x = -3.488$
$f(x) = 240.90741220147203$
Iteration 2:
$x = -3.159231053824$
$f(x) = 148.52441854620668$
Iteration 3:
$x = -2.9229164225026394$
$f(x) = 99.4029877988204$
Iteration 4:
$x = -2.742031675863951$
$f(x) = 70.0712149441725$
Iteration 5:
$x = -2.59779507407776$
$f(x) = 51.16573699678776$
Iteration 6:
$x = -2.4794003442716166$
$f(x) = 38.29644231132754$

Using 1200 iterations, we find a local minimum. Gradient descent converges after 251 iterations:

Iteration 246:
$x = -1.3972104037610875$
$f(x) = -4.348957706810287$
Iteration 247:
$x = -1.3972092332877633$
$f(x) = -4.348957708153158$
Iteration 248:
$x = -1.397208109187661$
$f(x) = -4.348957709391726$
Iteration 249:

$x = -1.3972070296234085$
$f(x) = -4.3489577105340995$
Iteration 250:
$x = -1.397205992830441$
$f(x) = -4.348957711587744$
Iteration 251:
$x = -1.397204997114115$
$f(x) = -4.348957712559557$
The local minimum occurs at -1.397204997114115.

(c) Starting from $x = 4$, using 6 iterations and a learning rate $\eta = 0.001$, we have:
Iteration 0:
$x = 4$
$f(x) = 198$
Iteration 1:
$x = 3.68$
$f(x) = 110.61233152000005$
Iteration 2:
$x = 3.450886144$
$f(x) = 64.53629857986431$
Iteration 3:
$x = 3.276396901609702$
$f(x) = 37.31076190742675$
Iteration 4:
$x = 3.138067975365072$
$f(x) = 19.971643359608052$
Iteration 5:
$x = 3.0252501730040535$
$f(x) = 8.322601113072949$
Iteration 6:
$x = 2.9312689375235244$
$f(x) = 0.17557478693807127$

Using 1200 iterations, we find a global minimum. Gradient descent converges after 171 iterations:

Iteration 166:
$x = 2.1472010808442814$
$f(x) = -26.611979763452368$
Iteration 167:
$x = 2.147199849708826$
$f(x) = -26.611979764921912$
Iteration 168:
$x = 2.147198693530893$
$f(x) = -26.61197976621797$

Iteration 169:
$x = 2.1471976077465786$
$f(x) = -26.611979767361007$
Iteration 170:
$x = 2.1471965880698725$
$f(x) = -26.611979768369096$
Iteration 171:
$x = 2.147195630475738$
$f(x) = -26.61197976925817$
The minimum occurs at 2.147195630475738.

(d) Starting from $x = -4$, using 1200 iterations and a learning rate $\eta = 0.01$, we have:

Iteration 0:
$x = -4$
$f(x) = 454$
Iteration 1:
$x = 1.12$
$f(x) = -8.71561728$
Iteration 2:
$x = 1.35166976$
$f(x) = -14.187225687602176$
Iteration 3:
$x = 1.588129914065571$
$f(x) = -19.554356180837104$
Iteration 4:
$x = 1.8001695002820235$
$f(x) = -23.55150883046352$
Iteration 5:
$x = 1.9599549783032466$
$f(x) = -25.64204722189585$
Iteration 6:
$x = 2.0585082124451546$
$f(x) = -26.383081197323108$
Iteration 7:
$x = 2.1089704990074423$
$f(x) = -26.568376620626637$
Iteration 8:
$x = 2.131573822006764$
$f(x) = -26.604622415479056$
Iteration 9:
$x = 2.140965264762093$
$f(x) = -26.6108073504332$
Iteration 10:
$x = 2.1447319392273982$

$f(x) = -26.611797434344084$
Iteration 11:
$x = 2.1462201881561778$
$f(x) = -26.611951695156172$
Iteration 12:
$x = 2.1468046545758837$
$f(x) = -26.611975468303953$
Iteration 13:
$x = 2.147033635512241$
$f(x) = -26.61197911612757$
Iteration 14:
$x = 2.147123260359083$
$f(x) = -26.611979674906664$
Iteration 15:
$x = 2.147158327192368$
$f(x) = -26.61197976044408$
Iteration 16:
$x = 2.147172045535117$
$f(x) = -26.611979773534642$
Iteration 17:
$x = 2.147177411923369$
$f(x) = -26.611979775537804$
Iteration 18:
$x = 2.147179511118853$
$f(x) = -26.611979775844326$
Iteration 19:
$x = 2.147180332263948$
$f(x) = -26.61197977589123$
The local minimum occurs at 2.147180332263948.

Using a larger learning rate, gradient descent converges faster. By doing bigger "jumps" from one value to the next, gradient descent takes a lower number of steps find the minimum and converges after 19 iterations vs. the 251 it used when η was 0.001.

(e) The algorithm fails at iteration 5 as the function approaches a too large number. To give an idea, when we start with:

$x_0 = -4$ and $f(x_0) = 454$,

$x_1 = x_0 - \eta f'(x_0) = -4 - 0.1 \cdot (-512) = 47.2$, $(f' = \frac{\partial f}{\partial x})$

and $f(47.2) = 9689505.955200002$.

In the next updates, the functions increases too much until approaching almost Inf:

Iteration 2:
$x = -82626.05440000002$
$f(x) = 9.321875746621314e + 19$
Iteration 3:
$x = 451278842347294.06$
$f(x) = 8.294875771953852e + 58$
Iteration 4:
$x = -7.352328532672759e + 43$
$f(x) = 5.8442611657954e + 175$

2 Problem 2

(a) Gradient Descent (GD) updates the weights for every example. So, we have 500 examples and 100 runs of GD (100 epochs), hence we update $v_{2,1}$ $500 \times 100 = 50,000$ times.

(b) i.
$$\begin{aligned}\Delta v_{2,3} &= -\eta \frac{\partial E}{\partial v_{2,3}} \\ &= -\eta \frac{\partial}{\partial v_{2,3}} \left\{ \frac{1}{2}[3(r_1 - y_1)^2 + 7(r_2 - y_2)^2] \right\} \\ &= -7\eta(r_2 - y_2)(-\frac{\partial y_2}{\partial v_{2,3}}) \\ &= 7\eta(r_2 - y_2)\frac{\partial}{\partial v_{2,3}}(v_2^T z) \\ &= 7\eta(r_2 - y_2)z_3.\end{aligned}$$

ii.
$$\begin{aligned}\Delta w_{h,j} &= \eta[3(r_1 - y_1)v_{1,h}z_h^{(1)}(1 - z_h^{(1)})x_j^{(1)} + \\ &+7(r_2 - y_2)v_{2,h}z_h^{(2)}(1 - z_h^{(2)})x_j^{(2)}].\end{aligned}$$

3 Problem 3

(a) Only **NeuralNetRK** because it is the only one that outputs any real numbers and does not use a function that bounds the output in $[0, 1]$.

(b) **NeuralNetCK**

	z_1	z_2	z_3	z_4	k_1	k_2	label
$x^{(1)}$	0	0	1	0	1	0	0
$x^{(2)}$	0	0	0	1	0	1	0

(c) **NeuralNetCK** because we have three classes that are mutually exclusive, and we need an output that represent the estimated probabilities for the three labels of x.

zeroone could be used, but we would need to add rules in case of ties, and it becomes complex when classes are exactly the same number.

(d) **NeuralNetRZeroOne** is a good candidate because we have two outputs that take values between 0 and 1. Though they are binary, and the model may give any real number between 0 and 1, this solution is preferable to NeuralNetCK because here the multiclass aren't mutually exclusive so don't necessarily need to sum to 1. In fact, a document can be about politics and be formal or informal, and a formal document can be about politics or not, etc.

$y_1, \ldots, y_K$ will satisfy $\sum_{i=1}^{K} y_i = 1$.

4 Problem 4

(a) Any decision tree algorithms, such as RF, the decision is a logical operation on one variable that can be either numerical or categorical. So, if we have logic rules like x=1, cut, or x=2 cut, and so forth, it would be the same as saying x="cat", cut, or x="dog" regardless of the ordinal value of number of alphabetical order of words. In NNet's, the activation functions are usually increasing or decreasing functions wrt their domain x (or linear combinations of x). So if x is ordered, it affects the decision making function output. Furthermore such functions only apply to numbers and don't have a domain defined as categories.

(b) i. Defining k_1 and k_2 where $k_1 = 1$ iff *stalk shape*="tapering" and $k_2 = 1$ iff *stalk shape*="enlarging", we have

ii. Because low, medium and high represent and ordered categorization. An item which is low is less than an item which is high wrt some measure. Therefore, when converting it into a number, the numeric input has to preserve the ordinal nature of the original attribute.

iii. For nominal attributes with only two values, it's generally fine to represent the two values as 0 and 1 (or -1 and +1) because it is not really relevant whether or not there is an ordinal relationship between

the two numbers as there are only two. Furthermore, using one-hot encoding (with disabled intercept to avoid collinearity) would not be different then representing the two values as 0 and 1 (and keeping the intercept).
For nominal attributes with multiple values, order becomes important so one-hot encoding is preferable for attributes that don't represent an order. For example, if we represent the examples [banana, apple, banana, pear, apple] into [1, 2, 1, 3, 2], we create an order leading to the fact that the average of banana and pear is apple.