CE4041 Sample Paper Solutions

Question 1

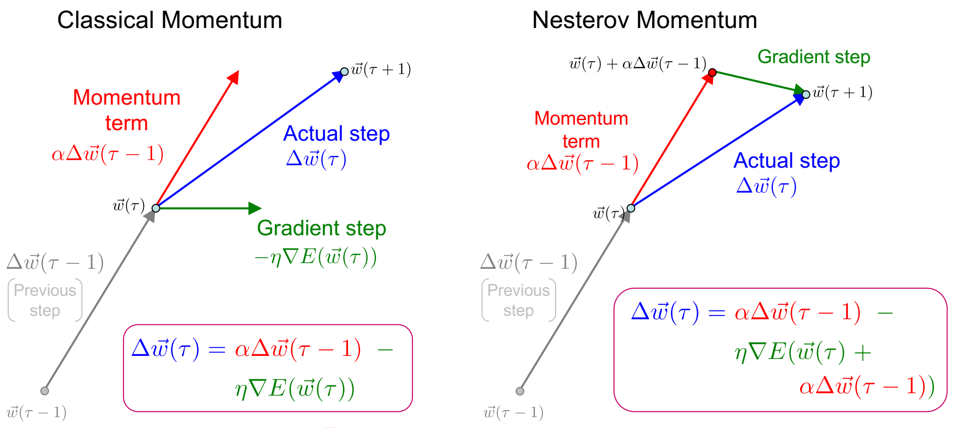
```
# Necessary imports
from tensorflow.keras.models import Sequential      # "layered" network
from tensorflow.keras.layers import Dense           # Fully-connected layer
from tensorflow.keras.layers import Conv2D          # 2-d Convolutional layer
from tensorflow.keras.layers import MaxPooling2D    # 2-d Max-pooling layer
from tensorflow.keras.layers import Flatten         # Convert 2-d layer 1-d
from tensorflow.keras.layers import Activation      # Nonlinearities

# Create a Keras model for a net.
#
# It has a single convolutional layer and a single maxpooling
# layer.  The net is then flattened and capped with a hidden
# layer and an output layer.
#
# The convolutional layer uses a 3 x 3 sampling window and has
# 10 slices (3 x 3 x 30).  Edges are padded with copies of the
# input data.
#
# Stride on the convolutional layer is *implicitly* 1.
#
# The Maxpool layer uses a 2 x 2 sampling window, without
# overlap (i.e., stides of 2 in both x and y).
#
# Finally the network is flattened to 1-d and fed to what is
# essentially a standard hidden+ouput backprop net for
# classification.

model = Sequential([
            Conv2D(10, kernel_size=3, padding='same',
           input_shape=training_images.shape[1:]),
            Activation('relu'),
            MaxPooling2D(pool_size=(2,2), strides=(2,2)),
            Flatten(),
            Dense(52),
            Activation('relu'),
            Dense(10),
            Activation('softmax')
        ])
```

Question 2

The diagram shows classical vs. Nesterov momentum. In this question classical momentum is used.

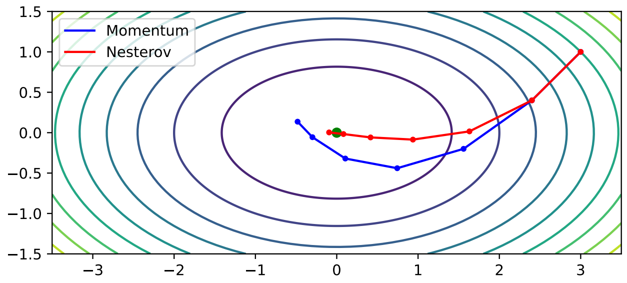

Classical Momentum
Momentum term
$\alpha\Delta\vec{w}(\tau-1)$
Actual step
$\Delta\vec{w}(\tau)$
$\vec{w}(\tau+1)$
$\vec{w}(\tau)$
Gradient step
$-\eta\nabla E(\vec{w}(\tau))$
$\Delta\vec{w}(\tau-1)$
Previous step
$\vec{w}(\tau-1)$
$\Delta\vec{w}(\tau) = \alpha\Delta\vec{w}(\tau-1) - \eta\nabla E(\vec{w}(\tau))$
Nesterov Momentum
$\vec{w}(\tau)+\alpha\Delta\vec{w}(\tau-1)$
Gradient step
$\vec{w}(\tau+1)$
Momentum term
$\alpha\Delta\vec{w}(\tau-1)$
Actual step
$\Delta\vec{w}(\tau)$
$\vec{w}(\tau)$
$\Delta\vec{w}(\tau-1)$
Previous step
$\vec{w}(\tau-1)$
$\Delta\vec{w}(\tau) = \alpha\Delta\vec{w}(\tau-1) - \eta\nabla E(\vec{w}(\tau) + \alpha\Delta\vec{w}(\tau-1))$

For this problem, the initial weight vector is $w_1 = (3.0, 1.0)$, and we assume that $\Delta\vec{w}(\tau = 0) = (0.0, 0.0)$ (makes sense as there is initially no previous update).

$\vec{w}(\tau)$ (w_1, w_2)	$\nabla E(\vec{w}(\tau))$ $(dE_n/dw_1, dE_n/dw_2)$	$\Delta\vec{w}(\tau)$ $(\Delta w_1, \Delta w_2)$	iteration (i.e., τ)
$(3.000, 1.000)$	$(6.000, 6.000)$	$(-0.600, -0.600)$	$\tau = 1$
$(2.400, 0.400)$	$(4.800, 2.400)$	$(-0.840, -0.600)$	$\tau = 2$
$(1.560, -0.200)$			$\tau = 3$

At the end of each cycle, $\Delta\vec{w}(\tau)$ is added to $\vec{w}(\tau)$ to make $\vec{w}(\tau+1)$. $\Delta\vec{w}(\tau)$ is formed from the scaled (η) gradient on step τ plus the scaled (α) $\Delta\vec{w}$ from the previous step.

Although not specifically asked in the question, it is interesting to see how classical momentum compares to Nesterov momentum for this problem (the latter is claimed to converge faster). Here, with 6 iterations, this is clearly the case. (Of course, you don't have to do this in an exam, it wasn't asked for there; but keep Nesterov in mind!)



Question 3

The general equation for W is

$$W(\vec{\lambda}) = \sum_i \lambda_i - 0.5 \sum_i \sum_j t_i t_j \lambda_i \lambda_j \vec{x}_i \vec{x}_j$$

For this problem the equation becomes

$$W(\lambda_1, \lambda_2) = \lambda_1 + \lambda_2 - 0.5 \sum_{i=1}^{2} \sum_{j=1}^{2} t_i t_j \lambda_i \lambda_j x_i x_i$$

or

$$W(\lambda_1, \lambda_2) = \lambda_1 + \lambda_2 - 0.5(\lambda_1^2 - 4\lambda_1\lambda_2 + 4\lambda_2^2)$$

Using the support vector machine constraint $\sum_i t_i \lambda_i = 0$, we have $-\lambda_1 + \lambda_2 = 0$, or $\lambda_1 = \lambda_2$, so W can be cast in terms of λ_1 alone as

$$W(\lambda_1) = 2\lambda_1 - 0.5\lambda_1^2$$

Finding the stationary point of this gives $\lambda_1 = 2$ (from $dW/d\lambda_1 = 2 - \lambda_1 = 0$). This is > 0, so a valid solution with all constraints active. $\lambda_2 = \lambda_1 = 2$.

Recover the weight w from $w = \sum \lambda_i t_i x_i = 2$.

Recover the bias b from either support vector $w_i x_i + b = t_i$. From $x_1 = 1$ we get $b = -3$. As a check, this works with $x_2 = 2$, $(2 \times 2 - 3 = 1)$.

Separating boundary is $wx - b = 0$, i.e., $2x - 3 = 0$, or $x = 1.5$, midway between the two input vectors (really, input points, as this is a 1-d problem), as expected.

Question 4

The search is admissible because $h(n) \leq h^*(n)$ for every node in the graph. On the other hand, the graph is not consistent because $h(A) = 7$, but $c(A, C) + h(C) = 4 + 1 = 5$, i.e.

$h(A) > c(A,C) + h(C)$ so $h(n) \leq c(n,n') + h(n')$ does not hold for every pair of states in the graph, violating the consistency criterion.

This means that we expect an A^* tree search to return an optimal solution, but an A^* graph search may return a suboptimal one.

Doing an A^* graph search, this involves using a closed set to track nodes already visited. This will only work is the problem is monotonic:

Agenda 1: $[[A]/7]$ (i.e., queue of nodes with f costs). Closed $=\{\}$.
Expand $[A]$. Successors of A: $\{C, B\}$.
$f(A \to C) = g(A \to C) + h(C) = 4 + 1 = 5$, $f(A \to B) = g(A \to B) + h(B) = 2 + 5 = 7$.

Agenda 2: $[[CA]/5, [BA]/7]$. Closed: $\{A\}$.
Expand $[CA]$. Successor of C: $\{D\}$.
$f(A \to C \to D) = g(A \to C \to D) + h(D) = 4 + 4 + 0 = 8$.

Agenda 3: $[[BA]/7, [DCA]/8]$. Closed: $\{A, C\}$.
Expand $[BA]$. Successor of B: $\{C\}$.
$f(A \to B \to C) = g(A \to B \to C) + h(C) = 2 + 1 + 1 = 4$.

Agenda 4: $[[CBA]/4, [DCA]/8]$. Closed: $\{A, B, C\}$.
$[CBA]$ *Eliminated* because C is in the closed set.

Agenda 5: $[[DCA]/8]$. Closed $=\{A, B, C\}$.
D is the goal, return path $A \to C \to D$, cost 8, suboptimal (lowest cost path is via B and costs 7).

Doing an A^* tree search, there is no closed set, so the search proceeds as follows:

Agenda 1: $[[A]/7]$.
Expand $[A]$. Successors of A: $\{C, B\}$.
$f(A \to C) = g(A \to C) + h(C) = 4 + 1 = 5$, $f(A \to B) = g(A \to B) + h(B) = 2 + 5 = 7$.

Agenda 2: $[[CA]/5, [BA]/7]$.
Expand $[CA]$. Successor of C: $\{D\}$.
$f(A \to C \to D) = g(A \to C \to D) + h(D) = 4 + 4 + 0 = 8$.

Agenda 3: $[[BA]/7, [DCA]/8]$.
Expand $[BA]$. Successor of B: $\{C\}$.
$f(A \to B \to C) = g(A \to B \to C) + h(C) = 2 + 1 + 1 = 4$.

Agenda 4: $[[CBA]/4, [DCA]/8]$.
Expand $[CBA]$. Successor of C: $\{D\}$.
$f(A \to B \to C \to D) = g(A \to B \to C \to D) + h(D) = 2 + 1 + 4 + 0 = 7$.

Agenda 5: $[[DCBA]/7, [DCA]/8]$.
D is the goal, return path $A \to B \to C \to D$, cost 7, optimal.

Question 5

```
Original formula is: ((b11 <=> (p12 | p21)) & !b11)

Converting to CNF by following the standard recipe:

Step 1. Eliminate all biconditionals.
  (((b11 => (p12 | p21)) & ((p12 | p21) => b11)) & !b11)

Step 2. Eliminate all implications.
  (((!b11 | (p12 | p21)) & (!(p12 | p21) | b11)) & !b11)

Step 3. Drive in negation so that it only governs atoms.
  (((!b11 | (p12 | p21)) & ((!p12 & !p21) | b11)) & !b11)

Step 4. Distribute OR over AND (may take several passes).
  Pass  1: (((!b11 | (p12 | p21)) & ((!p12 | b11) & (!p21 | b11))) & !b11)
Done

Final formula is: (!b11 & (!b11 | p12 | p21) & (!p12 | b11) & (!p21 | b11))

As set of clauses: {{!b11}, {!b11, p12, p21}, {!p12, b11}, {!p21, b11}}
```

Question 6

```
Conflict set, note representation of negation of conclusion
"!(c & d)" as the CNF clause "{!c, !d}", appended to the
clauses for the argument.

   1:  {!a, b}
   2:  {!b, c}
   3:  {a}
   4:  {!d}
   5:  {!c, !d}

       Successful clash of {!c, !d} against {!b, c} yielding {'!d', '!b'}.

   1:  {!a, b}
   2:  {!b, c}
   3:  {a}
   4:  {!d}
   5:  {!c, !d}
   6:  {'!d', '!b'}

       Successful clash of {'!d', '!b'} against {!a, b} yielding {'!d', '!a'}.

   1:  {!a, b}
   2:  {!b, c}
   3:  {a}
   4:  {!d}
   5:  {!c, !d}
   6:  {'!d', '!b'}
   7:  {'!d', '!a'}

       Clash of {'!d', '!a'} against {a} generated duplicate clause {'!d'}.
       Reject.
       No more clashes available for {'!d', '!a'}.  Backtracking.

   1:  {!a, b}
   2:  {!b, c}
   3:  {a}
   4:  {!d}
   5:  {!c, !d}
   6:  {'!d', '!b'}

       No more clashes available for {'!d', '!b'}.  Backtracking.

   1:  {!a, b}
```

```
2:  {!b, c}
3:  {a}
4:  {!d}
5:  {!c, !d}

    No more clashes available for {!c, !d}.  Backtracking.

    No (further) backtracking points available.
    FAILURE
```

It has not proved possible to derive an empty clause by resolution refutation. Since this is propositional logic, we can conclude that the formula $(c \wedge d)$ is therefore *not* entailed by the original clauses. (Note that we could not make this claim for predicate or higher-order logics, where we would simply have to be content with the statement "not proven".)