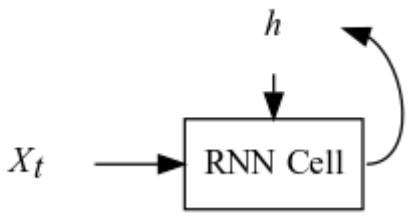
רשתות נשנות

בפרק זה נכיר את רכיב הרשת אשר יאפשר לרשת הנוירונים לקחת בחשבון את סדר הופעת המילים במשפט, זהו תא הרשת הנשנית (RNN cell) הקרוי לעתים גם תא אלמן (Elman cell) על שם החוקר הראשון לפרסם רשתות המבוססות עליו. בהמשך הדיון נניח כי משפט הקלט עבר טוקניזציה ושיכון, וכן ש- X_t הוא השיכון של הטוקן ה- t במשפט. הטוקנים מוזנים לפי הסדר אל תא הרשת הנשנית, בו ישנו מצב חבוי (hidden state) h אשר עובר עדכון עם הזנת כל טוקן. מלבד הטוקן, גם המצב החבוי הקודם משפיע על החישוב המתבצע בתא, וכך טוקנים המופיעים בתחילת המשפט משפיעים על המשך החישוב דרך השארת חותמם על המצב החבוי. להלן מצוייר היזון חוזר זה באופן סכמתי,



ביתר פרטים, בעת הזנת X_t לתא, המצב החבוי הבא מחושב ונשמר לפי הנוסחה

$$h_{t+1} = \tanh\left(W_{input}X_t + b_{input} + W_{hidden}h_t + b_{hidden}\right)$$

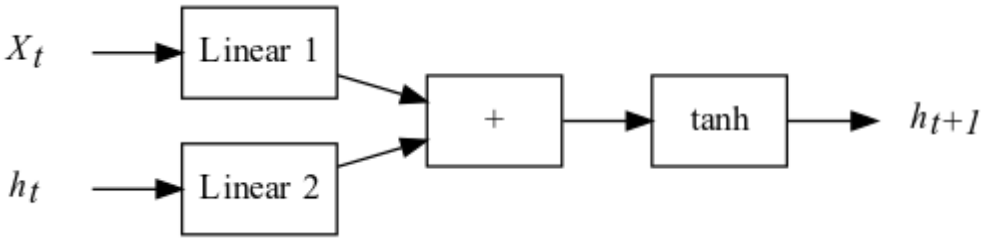
כאשר $W_{input}, b_{input}, W_{hidden}, b_{hidden}$ הם הפרמטרים הנלמדים של התא וכן

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

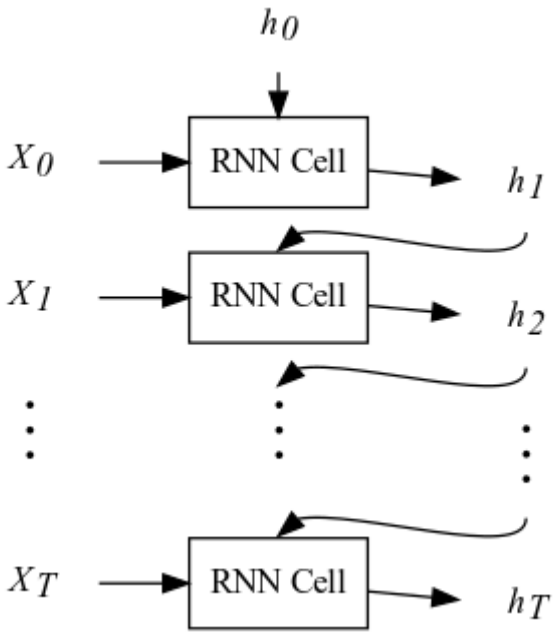
היא פונקציית הטנגנס ההיפרבולי: פונקציית אקטיבציה דמוית סיגמואיד אשר מחזירה ערכים בטווח $(-1,1)$. ניתן לפרק חישוב זה לשלבים:

1. הזנת טוקן הקלט (לאחר שיכון), X_t , לאגרגציה ליניארית בעלת הפרמטרים W_{input}, b_{input}.
2. הזנת המצב החבוי הקודם לאגרגציה ליניארית אחרת בעלת הפרמטרים W_{hidden}, b_{hidden}.
3. סכימת פלט שכבות האגרגציה.
4. הפעלת האקטיבציה על הסכום.
5. שמירת המצב החבוי החדש.

ובאיור,



בעת מעבר על דגימה אחת, משפט באורך T טוקנים, החישוב הנ"ל מתבצע בטור T פעמים, ואם נרצה לאייר את זרימת המידע ברשת במפורט, עלינו לקחת זאת בחשבון. ראו באיור הבא כיצד אנו פורסים (unfold) את איור התא המקורי לאורך מימד הזמן, ומביאים לידי ביטוי באופן מפורש את החישוב החוזר על עצמו. ודאו כי אתם מבינים שמדובר בתא נשנה יחיד, בעל אותם פרמטרים.



כעת נממש תא אלמן בקוד, כמודול של PyTorch ובהמשך נראה כיצד לשלבו ברשת נוירונים לחיזוי הרגש המובע במשפט.

```
class MyRNNCell(nn.Module):
    def __init__(self, embed_dim, hidden_dim):
        super().__init__()
        self.hidden_state = torch.zeros(hidden_dim)
        self.hidden_linear= nn.Linear(in_features = hidden_dim,
                                      out_features = hidden_dim)
        self.input_linear = nn.Linear(in_features = embed_dim,
                                      out_features = hidden_dim)
        self.activation   = nn.Tanh()
    def forward(self, one_embedded_token):
        Z1        = self.input_linear(one_embedded_token)
        Z2        = self.hidden_linear(self.hidden_state)
        Y         = Z1+Z2
        new_state = self.activation(Y)
        self.hidden_state = new_state
        #return
```

שימו לב לכך שהמצב החבוי מוגדר מראש להיות בעל `hidden_dim` מאפיינים ומאותחל באפסים, וכן שבעת מעבר קדימה יחיד מוזן לתא טוקן משוכן בגודל `embed_dim`. בשונה משכבות בהן נתקלנו עד כה – לא מוחזר כל ערך במעבר קדימה זה, אלא המצב החבוי עצמו מעודכן ונשמר בתא. ראו גם כיצד מוגדרות השכבות הליניאריות כך שפעולת הסכום תתבצע בין טנזורים בעלי אותו מימד, וכן שמימד המצב החבוי החדש יישאר זהה למימד המצב החבוי הקודם.

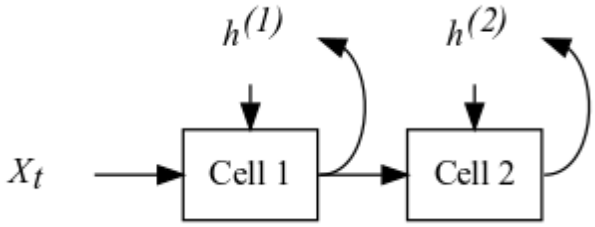
בבואנו לממש את הרשת הנשנית לסיווג המשפטים, נעשה מספר שינויים בתוך הרשת שכתבנו בפרק הקודם: נוותר על פעולת סכום השיכונים, אשר ביטלה את חשיבות סדר הופעתם במשפט, ובמקומה נזין את שיכוני הטוקנים לפי הסדר לתוך התא. על המצב החבוי האחרון, לאחר הזנת כל

הטוקנים, נחשוב בתור פלט מחלץ המאפיינים, ואותו נזין כרגיל לשכבה ליניארית ופונקציית ה-Softmax.

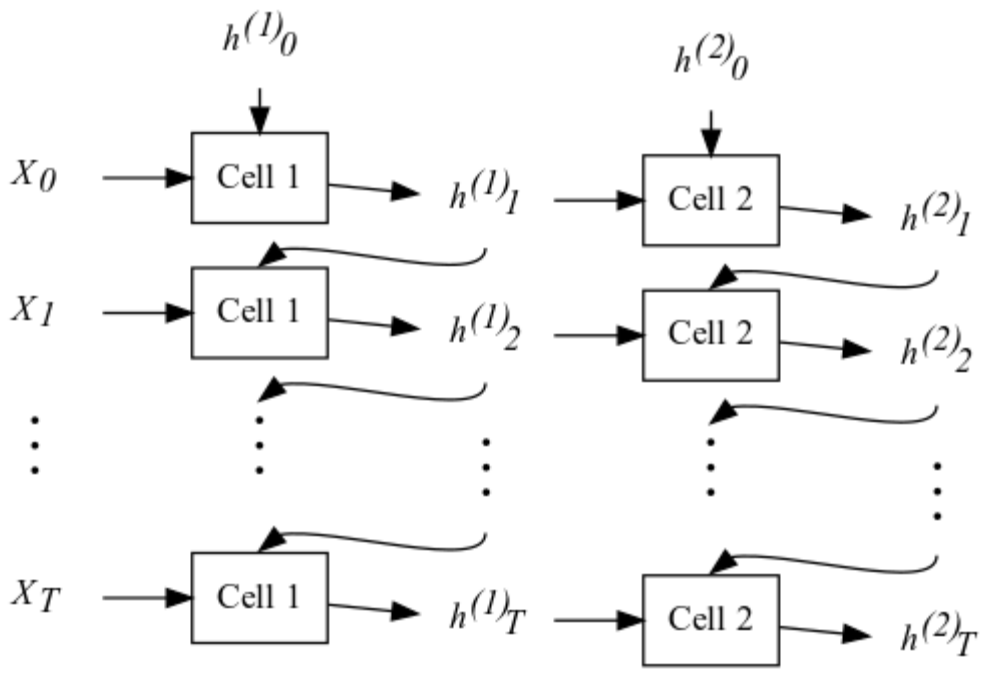
```
class RNNClassifier(nn.Module):
    def __init__(self, embed_dim, hidden_dim):
        super().__init__()
        self.hidden_dim = hidden_dim
        self.embedding  = nn.Embedding(len(vocab),embed_dim)
        self.rnn        = MyRNNCell(embed_dim, hidden_dim)
        self.linear     = nn.Linear(hidden_dim, 2)
        self.logsoftmax = nn.LogSoftmax(dim=0)

    def forward(self, sentence_tokens):
      self.rnn.hidden_state = torch.zeros(self.hidden_dim)
      for one_token in sentence_tokens:
        one_embedded_token = self.embedding(one_token)
        self.rnn(one_embedded_token)

      feature_extractor_output = self.rnn.hidden_state
      class_scores     = self.linear(feature_extractor_output)
      logprobs         = self.logsoftmax(class_scores)
      return logprobs
```

ראו כיצד הצורך בחישוב המצב החבוי החדש הבא על סמך המצב החבוי הקודם מוביל לשימוש בלולאה בעת המעבר קדימה ברשת: לפני חישוב המצב החבוי הסופי יש לחשב את כל הקודמים לו בסדרה.

כמו ברשתות בעלות קישוריות מלאה ורשתות קונבולוציה, ריבוי שכבות ועיבוד המידע לעומק השכבות מועיל גם עבור רשתות נשנות, אך במימוש הנ"ל אנו משתמשים רק בתא יחיד. כדי להרחיב את הרשת הנשנית לרשת עמוקה, עלינו לחבר תאי אלמן זה לזה, כך שהמצב החבוי של התא הקודם יועבר כקלט לתא הבא, במקומו של הטוקן המשוכן. נאייר זאת באופן סכמתי,



ואת החישוב המתבצע בתאים לאחר פריסה לאורך מימד הזמן,



כעת פלט מחלץ המאפיינים יהיה פלט התא **השני**, לאחר הזנת כל הטוקנים, המסומן ב- $h_T^{(2)}$ באיור הנ"ל. נממש רשת בעלות שתי שכבות נשנות על ידי שינוי קוד המודול הקודם במעט, כאשר השינויים מסומנים ב-#.

```
class DeepRNNClassifier(nn.Module):
    def __init__(self, embed_dim, hidden_dim):
        super().__init__()
        self.hidden_dim = hidden_dim
        self.embedding  = nn.Embedding(len(vocab),embed_dim)
        self.rnn1     = MyRNNCell(embed_dim, hidden_dim)
        self.rnn2     = MyRNNCell(hidden_dim, hidden_dim)       #
        self.linear      = nn.Linear(hidden_dim, 2)
        self.logsoftmax = nn.LogSoftmax(dim=0)

    def forward(self, sentence_tokens):
      self.rnn1.hidden_state = torch.zeros(self.hidden_dim)
      self.rnn2.hidden_state = torch.zeros(self.hidden_dim)    #
      for one_token in sentence_tokens:
        one_embedded_token = self.embedding(one_token)
        self.rnn1(one_embedded_token)
        self.rnn2(self.rnn1.hidden_state)                      #

      feature_extractor_output = self.rnn2.hidden_state        #
      class_scores      = self.linear(feature_extractor_output)
      logprobs          = self.logsoftmax(class_scores)
      return logprobs
```

שימו לב לכך שמימד הקלט של התא הנשנה השני הוא כמימד המצב החבוי, שכן המצב החבוי הראשון מהווה את הקלט עבורו.

במימוש הרשתות לעיל הדגשנו את בהירות החישוב המבוצע וזרימת המידע על פני יעילות זמן הריצה. על כן, לפני אימון הרשת, נחליף את השכבות הנשנות שלנו בערימה (stack) של RNN מובנות של PyTorch, המבצעות חישוב זהה אך בדרך יעילה יותר.

```
class FasterDeepRNNClassifier(nn.Module):
    def __init__(self, embed_dim, hidden_dim, RNNlayers):
        super().__init__()
        self.hidden_dim = hidden_dim
        self.embedding  = nn.Embedding(len(vocab),embed_dim)
        self.rnn_stack  = nn.RNN(embed_dim,                    #
                                 hidden_dim,
                                 RNNlayers)
        self.linear     = nn.Linear(hidden_dim, 2)
        self.logsoftmax = nn.LogSoftmax(dim=0)

    def forward(self, sentence_tokens):
      all_embeddings          = self.embedding(sentence_tokens)
      all_embeddings          = all_embeddings.unsqueeze(1)
      hidden_state_history, _= self.rnn_stack(all_embeddings)

      feature_extractor_output = hidden_state_history[-1,0,:]
      class_scores      = self.linear(feature_extractor_output)
      logprobs          = self.logsoftmax(class_scores)
      return logprobs
```

ראו כי אנו מעבירים את כל הטוקנים של המשפט בבת אחת לערימת ה-RNN, וברקע מתבצע החישוב בטור אך לא בלולאת פייתון. הערימה מחזירה כפלט את היסטוריית המצבים החבויים של התא האחרון בערימה, התא השני במימוש הקודם שלנו, ואנו שולפים מטנזור זה את האיבר האחרון כפלט מחלץ המאפיינים.

השכבות המובנות של PyTorch ערוכות לקבל minibatch של משפטים, כאשר באופן מקובל זהו המימד השני (המימד הראשון נשאר מימד הזמן) ולכן בעת המעבר קדימה הוספנו לשיכונים מימד מנוון בעזרת המתודה `unsqueeze`. הזנת הנתונים ב-batch דורשת עיבוד מקדים נוסף עקב אורכם המשתנה של המשפטים, בו לא נדון בקורס זה. יחד עם זאת, נוכל עדיין לאמן רשתות אלו כרגיל על ידי הזנת המשפטים אחד לאחר השני במחיר של תהליך אימון איטי יותר.

לרשתות נשנות יש את היכולת לחזות רגשות שונים עבור משפטים בעלי אותם טוקנים המופיעים בסדר שונה, בשונה מהרשת המבוססת על סכום שיכוני הטוקנים, אך כדי להשיג מטרה זו עלינו לאמנן בהצלחה. זהו אתגר לא פשוט אשר בסיבותיו נדון בפרק הבא.

שאלות לתרגול

1. אמנו RNN לחיזוי הרגש המובע במשפטים מאוסף הנתונים SST2 תוך שימוש בסט אימון קטן מאוד עד שתתקבל בבירור התאמת יתר. הראו זאת על ידי ציור גרף של שגיאת האימון ושגיאת הבדיקה.
2.
   א. מחקו את השורה הראשונה במעבר קדימה של רשת מסוג `RNNClassifier`, בה מאפסים את המצב החבוי לפני הזנת המשפט לרשת. כעת חזרו על השאלה הקודמת והשוו את התוצאות.
   ב. האם תוכלו להסביר את נחיצות האיפוס של המצב החבוי?

3. אמנו רשת RNN עמוקה תוך כדי שמירת נורמת אינסוף של גרדיאנט כל פרמטרי המודל.
ציירו בגרף את הנורמה הממוצעת של הגרדיאנטים כפונקציה של ה-Epoch.