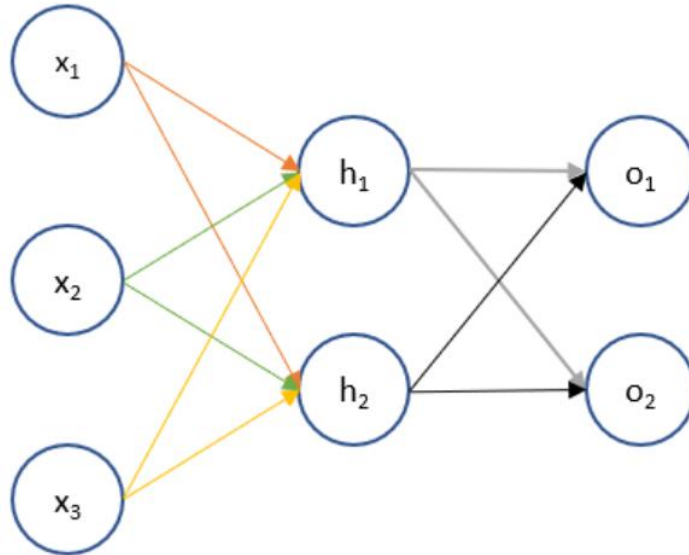


מבוא לבינה מלאכותית – תרגיל 3:

1. חישוב ידני של backpropagation:
נתונה רשת הניורונים הבאה:



עם bias (הוסיפו נירון שיזין את השכבות h ו- o),

פונקציית אקטיבציה של סיגמואיד: $\sigma(x) = \frac{1}{1+e^{-x}}$

ופונקציית שגיאה ריבועית: $E = \frac{1}{2}((t_1 - o_1)^2 + (t_2 - o_2)^2)$ (התיוג האמיתי לקלט מסומן

ב- (t_1, t_2)).

התחילו עם המשקלים הבאים:

$$\begin{aligned} w_1 = w_{x_1 \rightarrow h_1} &= 0.1, w_2 = w_{x_1 \rightarrow h_2} = 0.2, \\ w_3 = w_{x_2 \rightarrow h_1} &= 0.3, w_4 = w_{x_2 \rightarrow h_2} = 0.4, \\ w_5 = 0.5, w_6 = 0.6, b_1 &= 0.5, \\ w_7 = w_{h_1 \rightarrow o_1} &= 0.7, w_8 = 0.8, w_9 = 0.9, w_{10} = 0.1 \\ b_2 &= 0.5 \end{aligned}$$

נתונה נקודה $\vec{x} = (1, 4, 5)$, $\vec{t} = (0.1, 0.05)$

עדכנו את המודל ע"י מעבר פעם אחת של הנקודה על הרשת (השתמשו ב- gradient descent עם קצב למידה של 0.01). האם השתפרתם?

לשימושכם: $\sigma'(x) = \sigma(x)(1 - \sigma(x))$

2. למידה באמצעות רשת נוירונים, עצים, XGBoost ו-Random Forest: הערה- לתרגיל הזה אין מטרה מוגדרת, מעבר להיכרות עם המודלים השונים והמימוש שלהם בפייתון. עשו הכי טוב שתוכלו, בעזרת כל הכלים שלמדנו.
א. את הדאטא תשיגו בצורה הבאה:

```
from sklearn.datasets import load_breast_cancer  
  
x, y = load_breast_cancer(return_X_y=True)
```

ב. חלקו את הדאטא לקבוצות training ו-test באופן הבא (אל תשנו משתנים שנקבעו כאן):

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.18, random_state=42)
```

שימו לב, גם לחלוקה של ה- train כפי שמוסבר מטה כדאי להשתמש בפונקציה זו.

ג. עליכם להשתמש במודלים הבאים:

- a. רשת נוירונים – מוזמנים לממש אותה באמצעות [sklearn.neural_network.MLPClassifier](#) (פחות מומלץ כי מרחב האפשרויות לא הכי גדול שאפשר), או אחת מהחבילות keras (חבילה שבנויה על tensorflow ובעזרתה לימדתי בשנה שעברה. מוזמנים לחפש את התרגול המתאים משנה שעברה או להיעזר [במדריכים](#)) או pytorch (קישור [למדריכים](#)). יכול להיות שלמי שמתחיל בפייתון בכלל, החבילה הזאת פחות מתאימה).
- b. עץ החלטה בודד – מוזמנים להיעזר ב- [sklearn.tree.DecisionTreeClassifier](#).
- c. Random Forest – מוזמנים להיעזר ב- [sklearn.ensemble.RandomForestClassifier](#).
- d. XGBoost – מוזמנים להיעזר בחבילה [xgboost](#).

בחרו בעצמכם את ההיפר-פרמטרים המתאימים ביותר שתמצאו לכל מודל:
לרשת נוירונים – גדלי השכבות, כמות השכבות, פונקציית loss, מספר ה-epochs ועוד.
לעץ החלטה בודד – קריטריון החלוקה (gini impurity/information gain), עומק העץ המקסימלי, ההגבלות על מתי לחלק קבוצה ועוד.
ל- Random Forest – המאפיינים שקשורים לכל עץ בודד כמו למעלה, וכן מספר העצים שבהם נעשה שימוש.
ל- XGBoost – הפרמטרים הקשורים בעצים (max depth, subsample ועוד) וב- gradient boosting (לא למדנו איך זה מתבצע, אבל מופיעים שם פרמטרים שאחראיים לשליטה בכך, כמו קצב הלימוד eta ורגולריזציה lambda).

לא צריך להשתגע בחיפושם אחרי הפרמטרים.

המטרה: לקבל את ה- ROC AUC הגבוה ביותר המתקבל על ה-`test`. צרפו גם גרף של ה-`loss` כתלות ב-`epoch` בו אתם נמצאים, על ה-`train` ועל אחוז קטן שתחזיקו כ-`validation`.

את הפונקציה המחשבת את ה-AUC תשיגו ע"י הפקודה:
`from sklearn.metrics import roc_auc_score`

[קישור להסבר על AUC](#).

שימו לב: הגרפים של `loss` כתלות ב- `epoch` על ה- `train` לעומת על ה- `validation` נועדו לעזור לכם לבדוק מתי אתם `overfitting`. נסו להתמודד עם זה באמצעות שינויי הפרמטרים המתאימים. אתם אמורים להתקשות יותר בשליטה הזאת במודלים מבוססי העצים, אבל בכל זאת אפשר להגיע לערכים לא רעים על הדאטא הזה.

בהצלחה!