מבוא למערכות לומדות (236756)

סמסטר חורף תשע"ט

מבחן מסכם מועד ב' , 6 במרץ 2019

מספר סטודנט:

משך המבחן: 2.5 שעות. (150 דקות)

חומר עזר: אין להשתמש בכל חומר עזר. בעמוד הבא לרשותכם דף נוסחאות והגדרות.

הנחיות כלליות:

- המבחן כתוב בלשון זכר ומיועד לנשים ולגברים כאחד.
- מלאו את הפרטים בראש דף זה ובדף השער המצורף, בעט בלבד.
- במבחן 16 עמודים ממוספרים סהכ, כולל עמוד זה שמספרו 1. ודאו שיש לכם כל הדפים.
- במבחן 4 חלקים. יש לענות על **כל** השאלות.
- כל התשובות יכתבו על טופס הבחינה, **ויש להחזירו בתום הבחינה.**
- אנא כתבו בכתב יד קריא וברור. **תשובה בכתב יד שאינו קריא לא תיבדק.**
- נא לא לתלוש עמודים ממחברת הבחינה.
- נא לכתוב רק את מה שהתבקשתם ולצרף הסברים קצרים רק כפי שמבוקש בשאלה—אין צורך בהסברים או פרטים נוספים על אלו שהתבקשתם במפורש.

Less is More

בהצלחה!

דף נוסחאות

1. $\binom{n}{k} \le n^k$
2. שגיאת הכללה = true error = L_D^{01}.
3. שגיאת אימון = שגיאה אמפירית = training error = empirical error= L_S^{01} (ממוצע השגיאות על מדגם)
4. שגיאת הערכה =estimation error = $L_D^{01} - L_S^{01}$
5. $e \approx 2.72$
6. התפלגות גאומטרית עם פרמטר p

$$P(x=n) = p \quad (1-p)^{n-1}$$

7. תהא H מחלקת היפותזות של בעיית למידה כלשהי, ו- S קבוצת אימון שנבחרת באקראי. נסמן
$\hat{h} = argmin_{h\in H} L_D^{01}(h)$
$h \quad = argmin_{h\in H} L_S^{01}(h)$ אזי, לכל $\delta > 0$: בהסתברות של לפחות $1-\delta$ מתקיים:

$$L_D^{01}(\hat{h}) \le L_D^{01}(h \quad) + O\left(\sqrt{\frac{VCDIM(H) + \frac{1}{log(\delta)}}{|S|}}\right)$$

8. מאפיין = *feature*
9. סיווג = *label*
10. $[x]_{+=max\,(x,0)}$
11. מונום (*monomial*) במשתנים $x[1]$ ... $x[k]$ מדרגה d הוא מכפלה מהצורה $x[i_1]\cdot x[i_2]\cdots x[i_d]$ כאשר i_1 ... i_d הם בתחום $\{k..1\}$ (עם חזרות).
12. ⟦תנאי⟧ מוגדר כ *1* אם התנאי מתקיים, אחרת *0*.
13. *Decision stumps* על ייצוג מאפיינים כלשהו $x[1]..x[k]$ הוא מחלקת היפותזות הכוללת פונקציות מהצורה $h_{i,a} = [\![x[i] \ge a]\!]$ עבור *i=1..k* ו- a מספר ממשי כלשהו.

<u>חלק א : שאלות קצרות (30 נק')</u>

1. פונקציית השגיאה הריבועית $L(\hat{y}, y) = (\hat{y} - y)^2$ מהווה פונק שגיאה Surrogate עבור שגיאת ה 0-1
   - ☐ **אמת**
   - ☐ **שקר**

2. שגיאת ולידציה נמוכה מעידה תמיד על כך שהמודל יבצע באופן דומה על קבוצת המבחן
   - ☐ **אמת**
   - ☒ **שקר**

3. מקסימום של n פונקציות קמורות (ב R^d) היא פונקציה קמורה
   - ☒ **אמת**
   - ☐ **שקר** דוגמא נגדית (חובה לספק במקרה שסימנתם "שקר"):

4. פונקציית המטרה של Soft-SVM היא $\|w\|_2^2 + \lambda L_S^{hinge}(w)$
   ככל שמגדילים את λ, כך צפוי ששגיאת האימון (יש לסמן אפשרות אחת):
   - ☒ **תרד**
   - ☐ **תעלה**
   - ☐ **תישאר ללא שינוי**

5. אם מריצים אלגוריתם Halving (חצייה) במודל המקוון (on-line) על מחלקת היפותזה סופית, במקרה ה Realizable, אז מספר השגיאות שנעשה הוא סופי
   - ☐ **אמת**
   - ☐ **שקר**

6. הסבירו מהו Feature Selection מסוג wrapper

7. עבור פריור (prior) קבוע על מרחב פרמטרים, ככל שמספר הדוגמאות גדל משערך MAP בהכרח מתכנס למשערך MLE

- ☐ **אמת**
- ☐ **שקר**

8. אחרי מספר מספיק של איטרציות אלגוריתם AdaBoost יגיע לשגיאה 0 על קבוצת האימון

- ☐ **אמת** ✓
- ☐ **שקר** ✓

<u>חלק ב: סיבוכיות של מחלקות היפותזה</u>

תהי $\mathcal{H}$ מחלקת היפותזות בינאריות, מעל מרחב מדגם $\mathcal{X}$. תהי C תת קבוצה סופית של $\mathcal{X}$. לכל $h \in \mathcal{H}$ נסמן ב h_C את פונקציית ההגבלה (restriction) לקבוצה C, כלומר הפונקציה המוגדרת על התחום C ומזדהה עם הפונקציה h על תחום זה.

1. הגדירו <u>בצורה מדויקת</u> את פונקציית הגידול (growth function) $\tau_H: \{0,1,2 \ldots\} \mapsto \{0,1,2 \ldots\}$, כפי שנילמד בכיתה.

$\tau_H(m) =$

2. השלימו את ההגדרה של ניתוץ (shattering) של קבוצה.

קבוצה $C \subseteq \mathcal{X}$ מנותצת (shattered) על–ידי $\mathcal{H}$ אם לכל $y_1, \ldots, y_{|C|} \in \mathcal{Y}$ קיימת $h \in \mathcal{H}$
כך שלכל $i: 1, \ldots, c$ $h(x_i) = y_i$

3. הגדירו VC dimension

ממד ה VC ‏(VC dimension) של מחלקת ההיפותזות $\mathcal{H}$ הוא המספר המקסימלי d כך ש $l_S(h)=0$

4. ~~השלימו את משפט Sauer-Shelah~~

אם למחלקה $\mathcal{H}$ יש מימד VC סופי d, אז ______________________________

~~חלק~~ ג: מודלים הסתברותיים (27 נקודות)

1. בהינתן m הגרלות בת"ל $x_1..x_m$ הנדגמים מהתפלגות אחידה על $\{1,\ldots,N\}$ (כאשר N הוא פרמטר של ההתפלגות, מספר טבעי), חשבו את אומד הנראות המקסימלי (MLE) $\widehat{N}$:

2. נתון פריור $p(N)$ על הפרמטר N, כאשר p(N) פונקציה מונוטונית יורדת ממש כפונקציה של N. מהו משערך ה MAP ?

חלק ד: Stochastic Gradient Descent ו linear regression

1. השלימו את הטיעון הבא בהקשר של Stochastic Gradient Descent:

בהינתן בעיית מינימיזציה רב-מימדית $\min_{w \in R^d} f(w)$ מעל פונקציה f גזירה, פרוצדורת ה stochastic gradient descent מעדכנת בכל איטרציה t את הפיתרון הנוכחי w_t על ידי צעד מהצורה: $w_{t+1} \leftarrow w_t + v_t$, כאשר הווקטור v_t הוא משתנה מקרי המוגרל מהתפלגות שמקיימת:

$$E[v_t] = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_$$

2. בשאלה זו עליכם להשלים קוד שמממש Stochastic Gradient Descent עבור Linear Regression. רגרסיה לינארית היא בעיית המינימיזציה (ביחס למקדמים w) של

$$f(w) = \frac{1}{m}\sum_{i=1}^{m}(\langle x_i, w\rangle - y_i)^2$$

```
import numpy as np

# A function for running stochastic gradient descent
# on a linear regression problem
# x: Sample data of size [m,d], where m is the number
# of samples and d is the dimension.  The i'th data point
# equals x[i,:]
# y: labels (real numbers).  The i'th label is y[i]
# num_iter: Number of iterations to run
# step_size: Step size of gradient updates
#
```

```
def stochastic_gradient_descent_linear_regression(
    x, y, num_iters, step_size):

    # find the number of samples and the dimension
    m,d = np.size(x)

    # initialize w as zero weight vector
    w = np.zeros(d)

    for iter in range(_________________):

        # np.random.randint(N) returns a random
        # number in the range {0,1,…,N-1}
        i = np.random.randint(________________)

        ___________________________________

        ___________________________________

        ___________________________________

        ___________________________________

        for j in range(______________):

            w[j] = w[j] + ___________________________


    return w
```

3. בשאלה זו עליכם להשלים את קוד שמממש Stochastic Gradient Descent עבור Linear Regression עם רגולריזציה. רגרסיה לינארית עם רגולריזציה היא בעיית המינימיזציה (ביחס למקדמים w) של

$$f(w) = \frac{1}{m}\sum_{i=1}^{m}\left(\langle x_i, w\rangle - y_i\right)^2 + \lambda\|w\|_2^2$$

כאשר λ הוא פרמטר חיובי.

```
import numpy as np

# A function for running stochastic gradient descent
# on a regularized linear regression problem.
# x: Sample data of size [m,d], where m is the number
# of samples and d is the dimension.  The i'th data point
# equals x[i,:]
# y: labels (real numbers).  The i'th label is y[i]
# num_iter: Number of iterations to run
# step_size: Step size of gradient updates
# lambda: The lambda parameter from the problem
#         definition

def stochastic_gradient_descent_linear_regression(
     x, y, num_iters, step_size, lambda):

     # find the number of samples and the dimension
     m,d = np.size(x)

     # initialize w as zero weight vector
     w =np.zeros(d)

     for iter in range(_________________):
```

```
# np.random.randint(N) returns a random
# number in the range {0,1,…,N-1}
i = np.random.randint(________________)

___________________________________

___________________________________

___________________________________

___________________________________

for j in range(______________):

     w[j] = w[j] + ___________________________

return w
```

4. בסוף המבחן מצורף קוד של תכנית פייתון פשוטה. התוכנית מבצעת את הפעולות הבאות:
   a. מגרילה מטריצת נתונים X של 1000 שורות (דגימות) ו 10 עמודות (מימדים).
   כל קואורדינטה מוגרלת באופן בת"ל, מהתפלגות אחידה על הקטע [-0.5, 0.5]. נסמן ב x_i את הדגימה ה i (השורה ה i במטריצה).
   b. מגרילה וקטור מקדמים _w ממרחב 10 מימדי, כאשר כל קואורדינטה מוגרלת באופן בת"ל מהתפלגות אחידה על [-0.5,0.5].
   c. מייצרת סיווגים בינאריים של הדגימות, כאשר הסיווג של דגימה x_i הוא:
   $y_i = sign(\langle w\_, x_i \rangle)$
   d. מונה על גודל קבוצת אימון m, בין הערכים 10 ל 499 (כולל). עבור כל m:

i. מפצלת את הדגימות לקבוצת אימון (m הדגימות הראשונות) וקבוצת מבחן (1000-m הדגימות האחרונות)

ii. פותרת בעיית linear regression (כפי שהוגדר בשאלה 1, כלומר ללא רגולריזציה) ביחס לקבוצת האימון

iii. מחשבת את שגיאת ה linear regression על קבוצת האימון ועל קבוצת המבחן

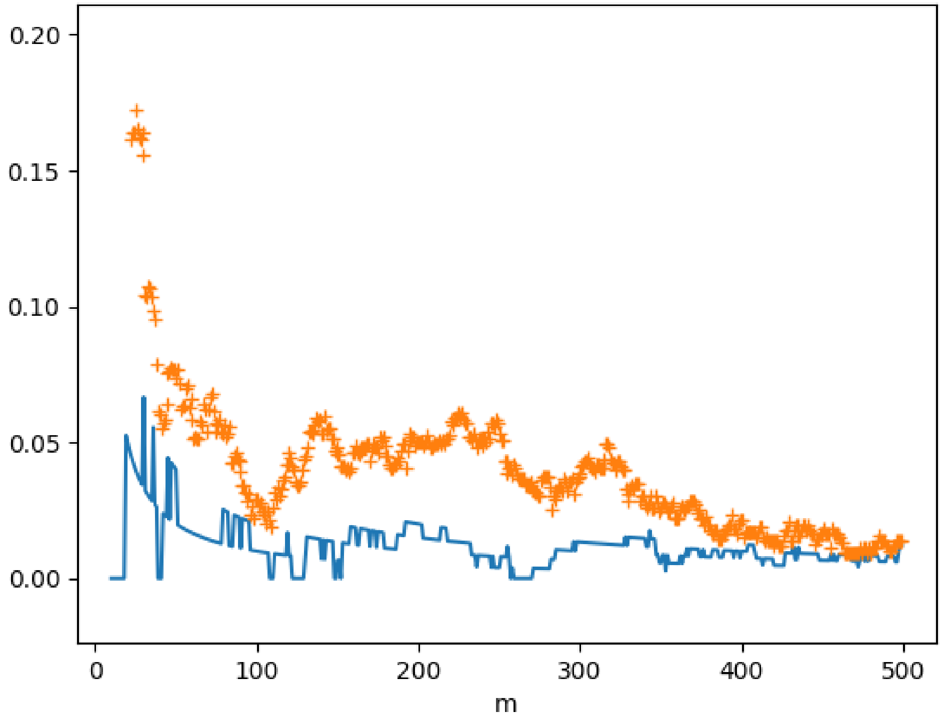
iv. מחשבת את שגיאת ה 0/1 loss על קבוצת האימון ועל קבוצת המבחן. להזכירכם, שגיאה זו על קבוצת נתונים $(x_1, y_1)..(x_m, y_m)$ מוגדרת כ $\frac{1}{m}\sum_{i=1}^{m}[\![y_i \neq sign(\langle w, x_i \rangle)]\!]$

e. משרטטת גרף של שגיאת האימון ושגיאת המבחן ביחס לפונקציית הרגרסיה הלינארית, כפונקציה של m

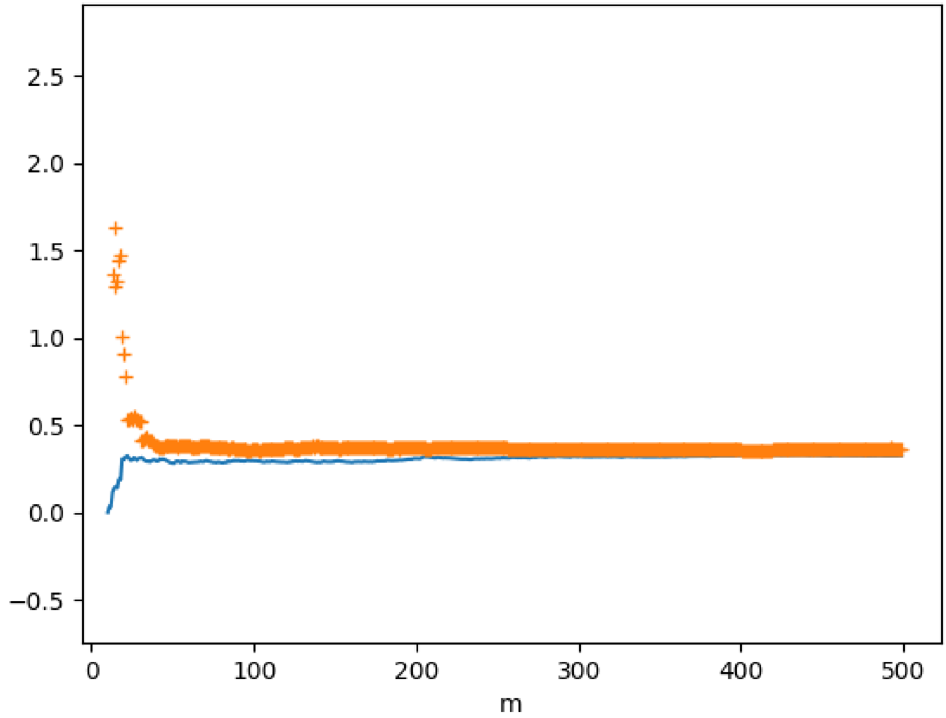
f. משרטטת גרף של שגיאת האימון ושגיאת המבחן ביחס לפונקצית ה 0/1 loss, כפונקצייה של m

להלן השירטוטים שפלטה התכנית. אחד השירטוטים מתאים ל regression loss והשני ל 0/1 loss:

שירטוט A :



שירטוט B:



סמנו את התשובה הנכונה: שירטוט A מתאים ל

- ☐ Regression loss
- ☐ 0/1 loss

בכל אחד מהשירטוטים, ישנו עקום אחד הבנוי מסימני '+' (markers) , והשני מנקודות '.'.
אחד העקומים מייצג את שגיאת האימון, והשני את שגיאת המבחן.

סמנו את התשובה הנכונה: העקומים המשורטטים ב '+' מייצגים את

- ☐ שגיאת האימון
- ☐ שגיאת המבחן

לנוחיותכם, אנו מצרפים את הקוד שייצר את השירטוטים, כפי שתואר באופן מילולי לעיל:

```
import numpy as np
import matplotlib.pyplot as plt
from my_ML_library import my_linear_regression_optimizer

# Generate random sample points: 1000 points of 10 dimensions
# Each coordinate uniform in range [-0.5,0.5]
X = np.random.random(size=(1000,10))-0.5

# Generate random weight vector: 10 coordinates uniform in range
# [-0.5,0.5]
w_ = np.random.random(size=(10,1))-0.5

# Generate labels y_i = sign(dot(x[i,:], w_))
y = np.sign(np.matmul(X, w_))

# Prepare vectors for saving error calculations for plot
train_errs_for_plot = []
test_errs_for_plot = []
train_errs_01_for_plot = []
test_errs_01_for_plot = []

# Enumerate over number of training points from 10 to 499
for m in range(10,500):

  # Slice data to train and test
  Xtrain = X[:m,:]
  Xtest = X[m:, :]
  ytrain = y[:m]
  ytest = y[m:]

  # Get optimal (column) coefficient vector
  w = my_linear_regression_optimizer(Xtrain, ytrain)
  w = w.reshape((10,1))

  # Compute training and test error (with respect to linear
```

```
  # regression cost)
  train_err = np.mean(np.square(np.matmul(Xtrain, w)-ytrain))
  test_err = np.mean(np.square(np.matmul(Xtest, w)-ytest))

  # compute training and test error (with respect to 0/1 loss)
  train_err_01 = \
    np.mean(np.sign(np.matmul(Xtrain, w)*ytrain) - 1.0) * (-0.5)
  test_err_01 = \
    np.mean(np.sign(np.matmul(Xtest, w)*ytest) - 1.0) * (-0.5)

  # Save error calculations for plot below
  train_errs_for_plot.append(train_err)
  test_errs_for_plot.append(test_err)
  train_errs_01_for_plot.append(train_err_01)
  test_errs_01_for_plot.append(test_err_01)

# Plot linear regression train and test errors, as a function
# of training set size m
plt.figure()
plt.xlabel('m')
plt.plot(range(10,500), test_errs_for_plot, [redacted])
plt.plot(range(10,500), train_errs_for_plot, [redacted])

# Plot 0/1 train and test errors, as a function of training
# set size m
plt.figure()
plt.xlabel('m')
plt.plot(range(10,500), test_errs_01_for_plot, [redacted])
plt.plot(range(10,500), train_errs_01_for_plot, [redacted])

plt.show()
```