Shelly Gamlielly — IML EX 3

Bayes Optimal and LDA

1) $\forall x \in X \quad h_D(x) = \begin{cases} 1 & P(Y=1 \mid X=x) \geq \frac{1}{2} \\ -1 & \text{else} \end{cases}$

$$h_D(x) = \underset{y \in \{-1,1\}}{\operatorname{argmax}} \; P(X=x \mid Y=y)\, P(Y=y) \qquad :B$$

במקרה A אנו מעוניינים לשערך את $P(Y=y \mid X=x)$
כך שבהינתן x, נוכל לחזות את ה- response
פונקציית הקלסיפיקציה מתארת את השגיאה רק $\ell(y,\hat{y}) = \mathbb{1}\{y \neq \hat{y}\}$

$$= \begin{cases} 1 & \text{if } y \neq \hat{y} \\ 0 & \text{else} \end{cases}$$

כאשר אנו חוזים $\hat{y}$ שונה מ-y
במקרה, אין שגיאה היא 0,
ואחרת השגיאה היא 1

אנו רוצים למזער את השגיאה $\min_{y \in \{+1,-1\}} P(Y \neq y \mid X=x)$
שזה שקול ל- $\max_{y \in \{1,-1\}} P(Y=y \mid X=x)$

כעת במקרה B לשערך את $P(Y=y \mid X=x)$
נשערך את $P(X=x \mid Y=y)$ כך שבהינתן ה-response
נגיע מה ההתפלגות של הX
ואת $P(Y=y)$ כלומר מה ההסתברות של כל קטגוריה

כעת לפי חוק בייס $P(Y=y \mid X=x) = P(X=x \mid Y=y)\dfrac{P(Y=y)}{P(X=x)}$

$$= \frac{P(X=x \mid Y=y)\,P(Y=y)}{\sum_{y \in Y} P(X=x \mid Y=y)\,P(Y=y)} = \frac{P(X=x \mid Y=y)\,P(Y=y)}{P(X=x \mid Y=1)P(Y=1) + P(X=x \mid Y=-1)P(Y=-1)}$$

כאשר, שזו מתאימה פונקציית הקלסיפיקציה כנדרשת
$Y = \{-1, 1\}$ ולכן

$$h_D(x) = 1 \iff \mathbb{P}(Y=1 \mid X=x) > \mathbb{P}(Y=-1 \mid X=x)$$

$$\iff \frac{\mathbb{P}(X=x \mid Y=1)\,\mathbb{P}(Y=1)}{\mathbb{P}(X=x)} > \frac{\mathbb{P}(X=x \mid Y=-1)\,\mathbb{P}(Y=-1)}{\mathbb{P}(X=x)}$$

$$\iff \mathbb{P}(X=x \mid Y=1)\,\mathbb{P}(Y=1) > \mathbb{P}(X=x \mid Y=-1)\,\mathbb{P}(Y=-1)$$

ובאותו אופן

$$h_D(x) = -1 \iff \mathbb{P}(X=x \mid Y=-1)\,\mathbb{P}(Y=-1) > \mathbb{P}(X=x \mid Y=1)\,\mathbb{P}(Y=1)$$

כעת לאחר השוויוניות הללו, נראה שזה מתיישב עם ההגדרה
של $h_D(x)$ שמחזיר את הערך עם ההסתברות הגבוהה יותר.

$$h_D(x) = 1 \iff \mathbb{P}(Y=1 \mid X=x) \geq \frac{1}{2} \iff$$

$$\frac{\mathbb{P}(X=x \mid Y=1)\,\mathbb{P}(Y=1)}{\mathbb{P}(X=x)} \geq \frac{1}{2} \iff$$

$$\mathbb{P}(X=x \mid Y=y)\,\mathbb{P}(Y=1) \geq \frac{\mathbb{P}(X=x \mid Y=1)\,\mathbb{P}(Y=1) + \mathbb{P}(X=x \mid Y=-1) \cdot \mathbb{P}(Y=-1)}{2}$$

$$\iff \frac{\mathbb{P}(X=x \mid Y=1)\,\mathbb{P}(Y=1)}{2} \geq \frac{\mathbb{P}(X=x \mid Y=-1)\,\mathbb{P}(Y=-1)}{2}$$

$$\iff \mathbb{P}(X=x \mid Y=1)\,\mathbb{P}(Y=1) > \mathbb{P}(X=x \mid Y=-1)\,\mathbb{P}(Y=-1)$$

כעת באותו אופן נקבל

$$h_D(x) = -1 \iff \mathbb{P}(Y=1 \mid X=x) < \frac{1}{2}$$

$$\iff \mathbb{P}(X=x \mid Y=1)\,\mathbb{P}(Y=1) < \mathbb{P}(X=x \mid Y=-1)\,\mathbb{P}(Y=-1)$$

(כנדרש)

MAP estimation - classification problem

$$\min_{x \in \mathcal{X}} \mathbb{P}(X \neq x \mid Y = y) \quad \forall y \in Y$$

שקול ל-

$$\max_{x \in X} \mathbb{P}(X = x \mid Y = y) \quad \forall y \in Y$$

במקרה בו Y רציף

$$\mathbb{P}(X = x \mid Y = y) = \frac{\mathbb{P}(X = x)}{f_Y(y)} f_{Y \mid X = x}(y) \propto \mathbb{P}(X = x) f_{Y \mid X = x}(y)$$

נשים לב שנרצה למקסם את המונה ביחס ל-X, המכנה קבוע לכל $y \in Y$
ולכן זה שקול ל

$$\max_{x \in X} \mathbb{P}(X = x) f_{Y \mid X = x}(y)$$

במקרה בו $X = \{x_1, x_2\}$ בינארי binary classification מתקיים כי
המסווג האופטימלי הוא $\hat{X}^*(y) = x_1 \iff$

$$\mathbb{P}(X = x_1) f_{Y \mid X = x_1}(y) > \mathbb{P}(X = x_2) f_{Y \mid X = x_2}(y)$$

2) $$\mathbb{P}(Y=y \mid X=x) = \frac{f_{X|Y}(x|y)\,\mathbb{P}(Y=y)}{\mathbb{P}(X=x)}$$

נבחין כי המכנה לא תלוי ב-y, ולכן מעוניינים למקסם את
המונה כתלות ב-y ולכן

נסמן $\frac{1}{\mathbb{P}(X=x)} = C$

$$= C\, f_{X|Y}(x|y)\,\mathbb{P}(Y=y)$$

נציב ונקבל

$$= C \cdot \mathbb{P}(Y=y)\,(2\pi)^{-\frac{d}{2}}\,|\det\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu_y)^T\Sigma^{-1}(x-\mu_y)\right)$$

כעת נתייחס לכל הגורמים שלא תלויים ב-y ונסמן ב-

$$C' = (2\pi)^{-\frac{d}{2}}\,|\det\Sigma|^{-\frac{1}{2}} \cdot C$$

ולכן

$$\mathbb{P}(Y=y \mid X=x) = C'\,\mathbb{P}(Y=y)\exp\left(-\frac{1}{2}(x-\mu_y)^T\Sigma^{-1}(x-\mu_y)\right)$$

נפעיל log ונקבל

$$\log(\mathbb{P}(Y=y \mid X=x)) = \log C' + \log\mathbb{P}(Y=y) - \frac{1}{2}(x-\mu_y)^T\Sigma^{-1}(x-\mu_y)$$

($\log C'$ לא תלוי ב-y)

נקבל

$$\underset{y\in\{-1,1\}}{\operatorname{argmax}} \log\mathbb{P}(Y=y) - \frac{1}{2}(x-\mu_y)^T\Sigma^{-1}(x-\mu_y)$$

$$= \underset{y\in\{-1,1\}}{\operatorname{argmax}} \log\mathbb{P}(Y=y) - \frac{1}{2}\left[x^T\Sigma^{-1}x + \mu_y^T\Sigma^{-1}\mu_y\right] + x^T\Sigma^{-1}\mu_y$$

$$= \underset{y\in\{-1,1\}}{\operatorname{argmax}} C'' + \log(\mathbb{P}(Y=y)) - \frac{1}{2}\mu_y^T\Sigma^{-1}\mu_y + x^T\Sigma^{-1}\mu_y$$

נסמן $C'' = -\frac{1}{2}x^T\Sigma^{-1}x$

$$= \underset{y\in\{-1,1\}}{\operatorname{argmax}} \ln\mathbb{P}(Y=y) - \frac{1}{2}\mu_y^T\Sigma^{-1}\mu_y + x^T\Sigma^{-1}\mu_y$$

$$= \underset{y\in\{-1,1\}}{\operatorname{argmax}} \delta_y(x)$$

כלומר נתון הקלט x, ניתן לחזות את ה-response ע"פ
ה-$\delta_y(x)$ הגבוה יותר

3) Write a formula to estimate μ_{+1}, μ_{-1} and $\mathbb{P}(y)$ based on training set $S = (x_1,y_1), \dots (x_m,y_m)$

$$\mathbb{P}(Y=y) = \frac{|\{i : y_i = y\}|}{m} = \frac{\text{מס' הדוגמאות עם תיוג } y}{\text{מס' הדוגמאות סה"כ}}$$

וקטור תוחלות
$$\mu_y = \frac{1}{|\{i : y_i = y\}|} \sum_{i : y_i = y} x_i$$
ממוצע הדוגמאות x_i שתיוגם שלהן הוא y

מטריצת הקווריאנס של התפלגויות השוות
$$\Sigma = \frac{1}{m-2} \sum_{y \in \{\pm 1\}} \sum_{i : y_i = y} (x_i - \mu_y)^2 = \frac{1}{m-2} \sum_{y \in \{\pm 1\}} \sum_{i : y_i = y} (x_i - \mu_y)(x_i - \mu_y)^T$$

(ההסתברות מתקבלת ע"י גזירה חלקית כמו פעם עם הפונקציה הרלוונטית והשוואה ל-0, נגזור ע"פ הפיתוח של שאלה 2 שם הוצאנו log על שני צידי המשוואה)

4) המסווג מקבל מתוכן קלט מייל ומחליט האם מדובר בהודעת spam או לא

שתי סוגי השגיאות שהמסווג יכול לעשות:

1) המייל היה הודעת spam, אך המסווג החליט שאינה spam
2) המייל לא היה הודעת spam, אך המסווג החליט שמדובר בהודעת spam

לאלו מתאים שגיאות מסוג: False positive, False Negative

השגיאה שלדעתי גרועה יותר היא שה-classifier החליט שההודעה שאינה spam היא הודעת spam. וכך, המשתמש יכול לפספס הודעה חשובה.

Type I error הוא מסוג False Positive כלומר ה-true label הוא $y = \{\text{not spam}\}$ ו- $\hat{y} = \{\text{spam}\}$

← spam - positive label, not spam - negative label

נסמן $y = -1$ ה-label הוא המייל אינו spam

$y = 1$ ה-label הוא המייל הוא spam

נקבל מייל רגיל ונסווג אותו כ-spam. שגיאה כזו, שלנו עבור:

$y = -1$, $\hat{y} = 1$

לא שגיאה יותר חמורה שיעבור לתיבת ה-inbox מייל של spam: $y = 1$, $\hat{y} = -1$

	$y=1$	$y=-1$
$\hat{y}=1$	TP	FP (Type I error)
$\hat{y}=-1$	FN Type II error	TN

SVM - Formulation

5) QP: $\underset{v \in \mathbb{R}^n}{\operatorname{argmin}} \ \frac{1}{2} v^T Q v + a^T v$

$$\text{s.t} \quad Av \leq d$$

where $Q \in \mathbb{R}^{n \times n}$, $a \in \mathbb{R}^n$, $d \in \mathbb{R}^m$, $A \in \mathbb{R}^{n \times m}$

Write the Hard SVM problem as a QP problem in canonical form.
What are the values of Q, A, a, d that express this problem as QP in canonical form?

$$Q = 2 I_{n \times n} = \begin{pmatrix} 2 & & & \\ & 2 & & \\ & & \ddots & \\ & & & 2 \end{pmatrix}, \quad w = (w_1, \dots, w_n)^T$$

$$v = (w, b)^T, \quad a^T = 0, \quad d = \begin{pmatrix} -1 \\ \vdots \\ -1 \end{pmatrix}$$

$$A = -\begin{pmatrix} y_1 x_{11} & y_1 x_{12} & \cdots & y_1 x_{1n} & y_1 \\ y_2 x_{21} & y_2 x_{22} & \cdots & y_2 x_{2n} & y_2 \\ \vdots & & & & \\ y_m x_{m1} & y_m x_{m2} & \cdots & y_m x_{mn} & y_m \end{pmatrix}$$

$\underset{(w,b)}{\operatorname{argmin}} \ \|w\|^2$ Hard SVM.

$$\text{s.t} \quad \forall i \ \ y_i(\langle w, x_i \rangle + b) \geq 1$$

$$\frac{1}{2} v^T Q v + a^T v = \frac{2}{2} v^T I v + 0$$

$$= v^T v = w^T w = \|w\|^2$$

$\underset{v \in \mathbb{R}^n}{\operatorname{argmin}} \ \frac{1}{2} v^T Q v + a^T v \Rightarrow \underset{(w,b)}{\operatorname{argmin}} \|w\|^2$

$$Av \leq d \Rightarrow -\begin{pmatrix} y_1 x_{11} & y_1 x_{12} & \cdots & y_1 x_{1n} & y_1 \\ \vdots & & & & \\ y_m x_{m1} & y_m x_{m2} & \cdots & y_m x_{mn} & y_m \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{pmatrix} \leq -1$$

$1 \leq i \leq n$ לכל

$$-(y_i x_{i1} \ \ y_i x_{i2} \ \cdots \ y_i x_{in} \ y_i) \begin{pmatrix} w_1 \\ \vdots \\ w_n \\ b \end{pmatrix} = -y_i \langle x_i, w \rangle - y_i b$$

$$= -y_i(\langle x_i, w \rangle + b) \leq -1$$

$$\Rightarrow y_i(\langle x_i, w \rangle + b) \geq 1$$

6) Soft SVM: $\underset{w, \{\xi_i\}}{\operatorname{argmin}} \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} \xi_i$

$$\text{s.t } \forall i \quad y_i \langle w, x_i \rangle \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0$$

Show that this problem is equivalent to

$$\underset{w}{\operatorname{argmin}} \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} \ell^{hinge}(y_i \langle w, x_i \rangle)$$

where $\ell^{hinge}(a) = \max\{0, 1-a\}$

$$\ell^{hinge}(y_i \langle w, x_i \rangle) = \max\{0, 1 - y_i \langle w, x_i \rangle\}$$

נשים לב כי $y_i \langle w, x_i \rangle \geq 1 - \xi_i \iff$

$1 - y_i \langle w, x_i \rangle \leq \xi_i$

מכאן, שאם נחליף את האילוצים:

$\forall i \quad y_i \langle w, x_i \rangle \geq 1 - \xi_i \; , \; \xi_i \geq 0$

לכן $\forall i \quad 1 - y_i \langle w, x_i \rangle \leq \xi_i \; , \; \xi_i \geq 0$

לכן $\forall i \quad \xi_i \geq \max\{0, 1 - y_i \langle w, x_i \rangle\}$

$= h^{hinge}(y_i \langle w, x_i \rangle)$

נניח בשלילה כי $\xi_i > \max\{0, 1 - y_i \langle w, x_i \rangle\}$

יהי $\{\xi_i\}, w$ פתרון למע"א האופטימיזציה

כלומר נניח שקיים k כך ש $\xi_k > \max\{0, 1 - y_k \langle w, x_k \rangle\}$

נגדיר $\xi'_i = \max\{0, 1 - y_i \langle w, x_i \rangle\}$

ולכן $\xi_i > \xi'_i \Leftarrow$ $\frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} \xi_i$

בסתירה לכך שהוא מינימלי $> \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} \xi'_i$

אנו פותרים בעיית אופטימיזציה והמטרה היא למצוא את המינימום

הפתרון כי קיים ξ'_i שבו ניתן תוצאה קטנה יותר, בסתירה לאופטימליות של ξ_i

כלומר הוכחנו שבהכרח ולכן $\xi_i = \max\{0, y_i \langle w, x_i \rangle\}$

הערה: הראינו בשלב הראשון עם הגדרה כי

$$\xi_i \geq \max\{0, 1 - y_i \langle w, x_i \rangle\}$$

ולא הראינו כי $\xi_i = \max\{0, 1 - y_i \langle w, x_i \rangle\}$ אך כן שהוכחנו בשלילה

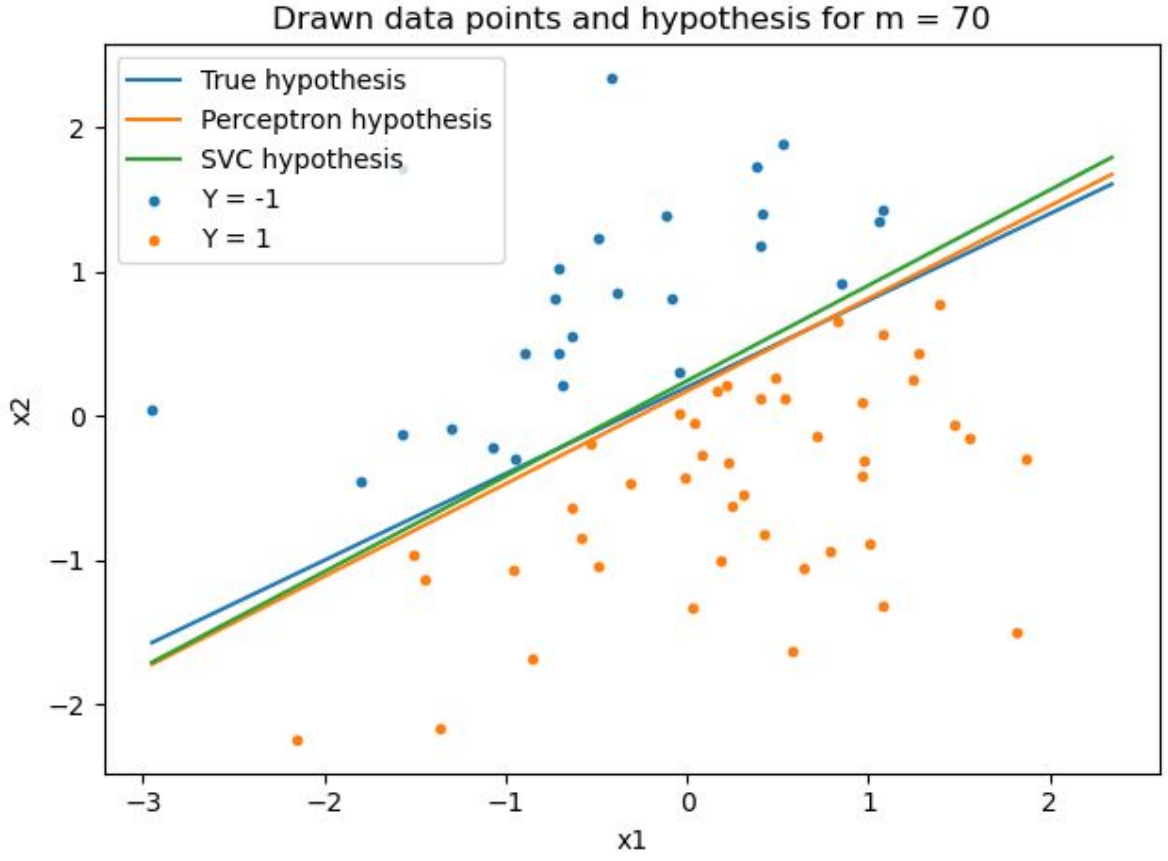
כי $y_i > \max\{0, 1 - y_i \langle w, x_i \rangle\}$ והגענו לסתירה.

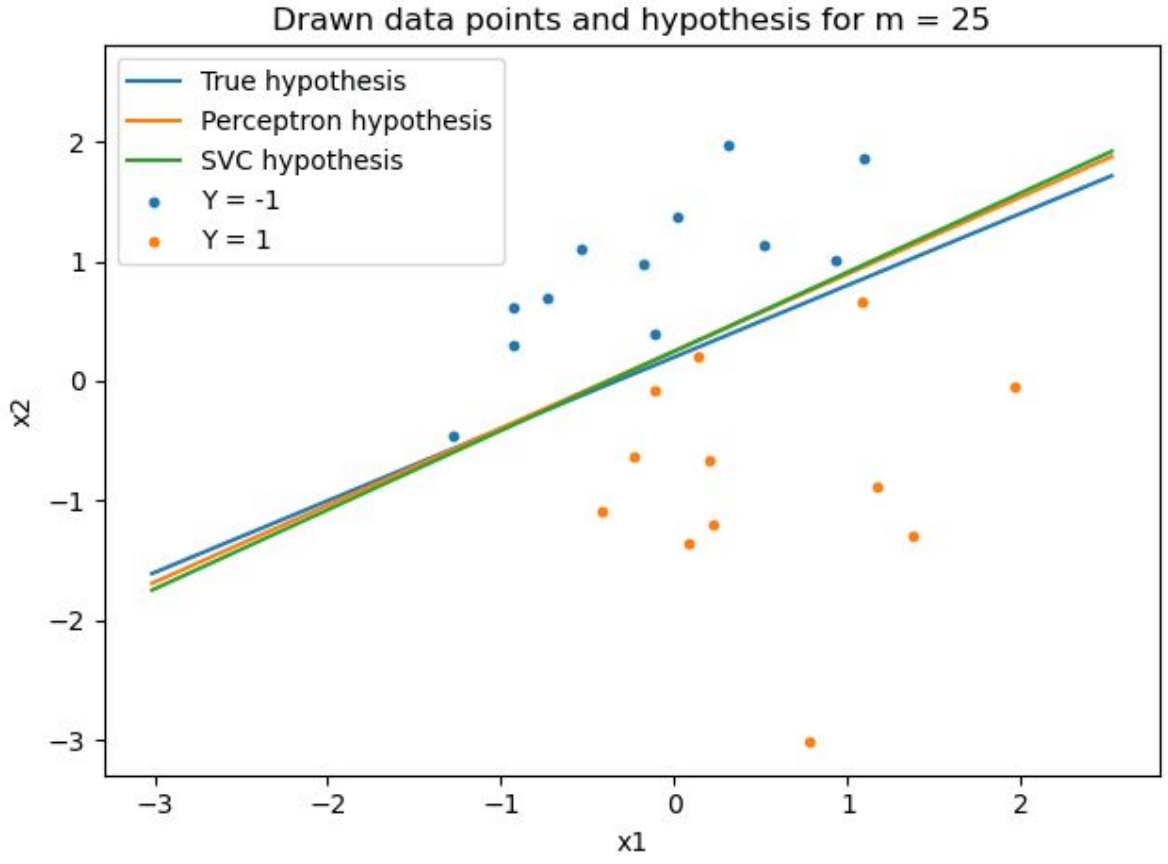
Iml Ex3

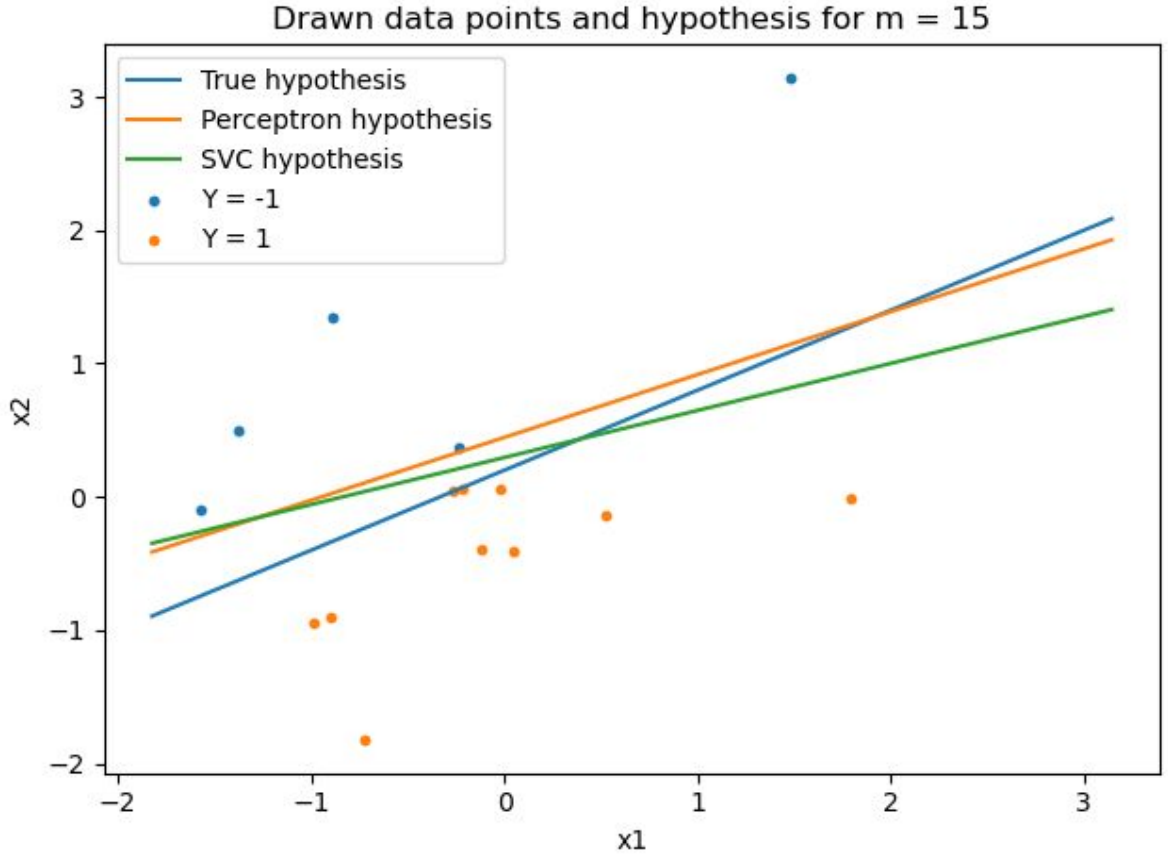
9. For each $m \in \{5, 10, 15, 25, 70\}$, draw m training points $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and create the following figure:

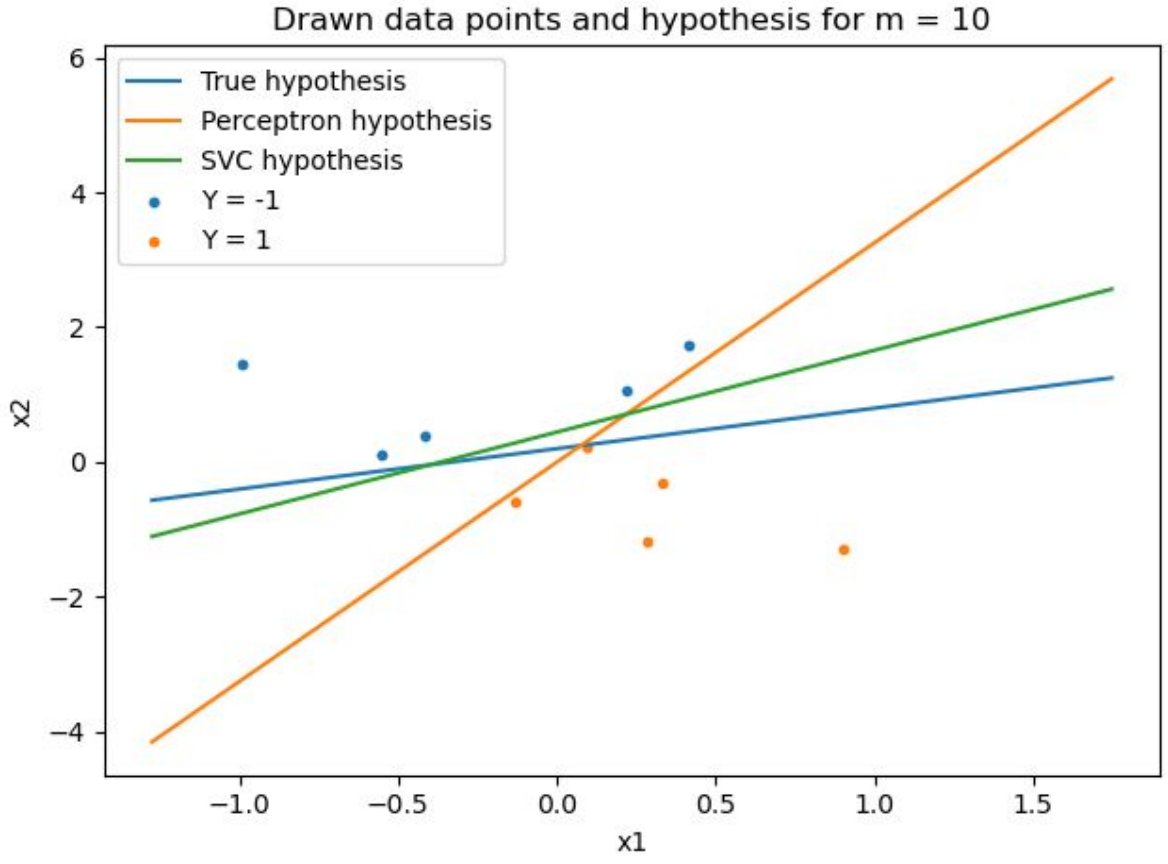
   - The drawn data points, colored according to their labels (e.g., blue for positive labels and orange for negative ones)
   - Add the hyperplane of the **true hypothesis** (the function f)
   - Add the hyperplane of the hypothesis generated by the **perceptron**
   - Add the hyperplane of the hypothesis generated by **SVM**
   - Add a legend to explain which hyperplane is which

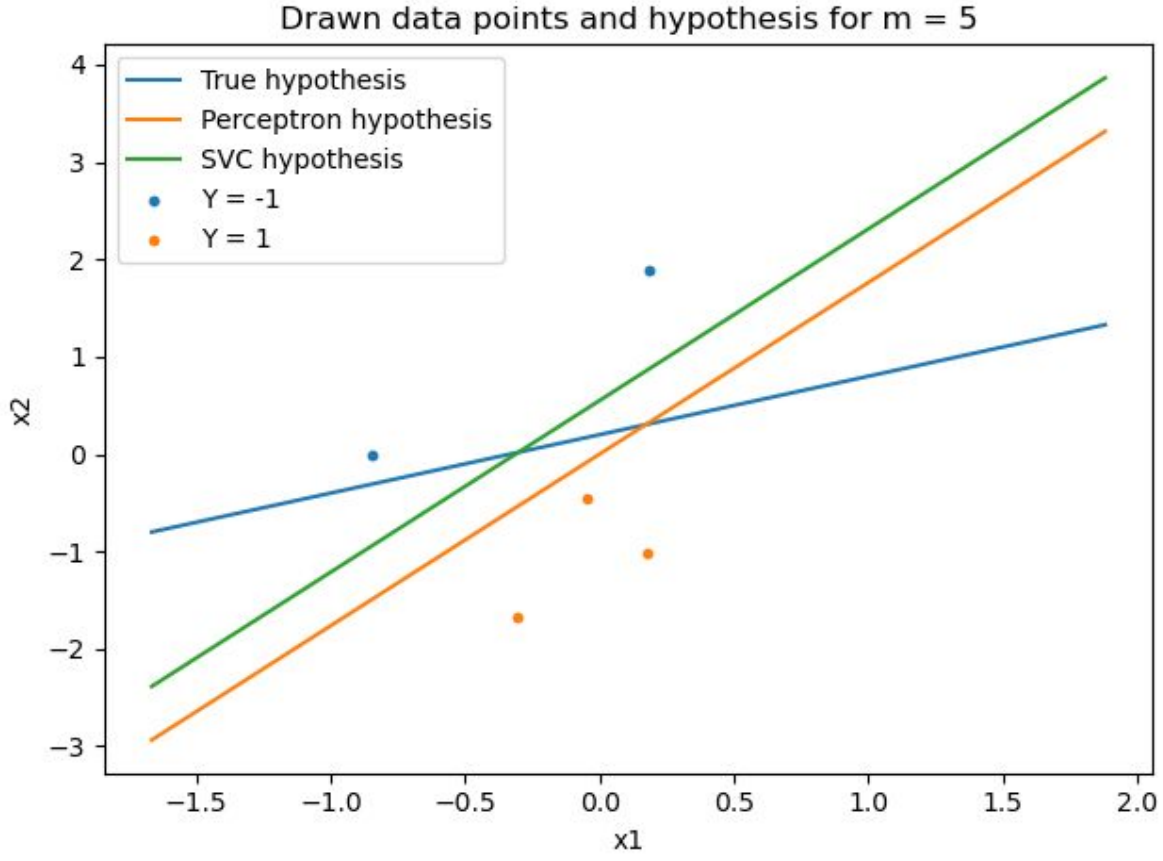
   Add the 5 plots (one for each m) to your PDF file.


Drawn data points and hypothesis for m = 70
True hypothesis
Perceptron hypothesis
SVC hypothesis
Y = -1
Y = 1
x1
x2


Drawn data points and hypothesis for m = 25
True hypothesis
Perceptron hypothesis
SVC hypothesis
Y = -1
Y = 1
x1
x2


Drawn data points and hypothesis for m = 15
True hypothesis
Perceptron hypothesis
SVC hypothesis
Y = -1
Y = 1
x1
x2

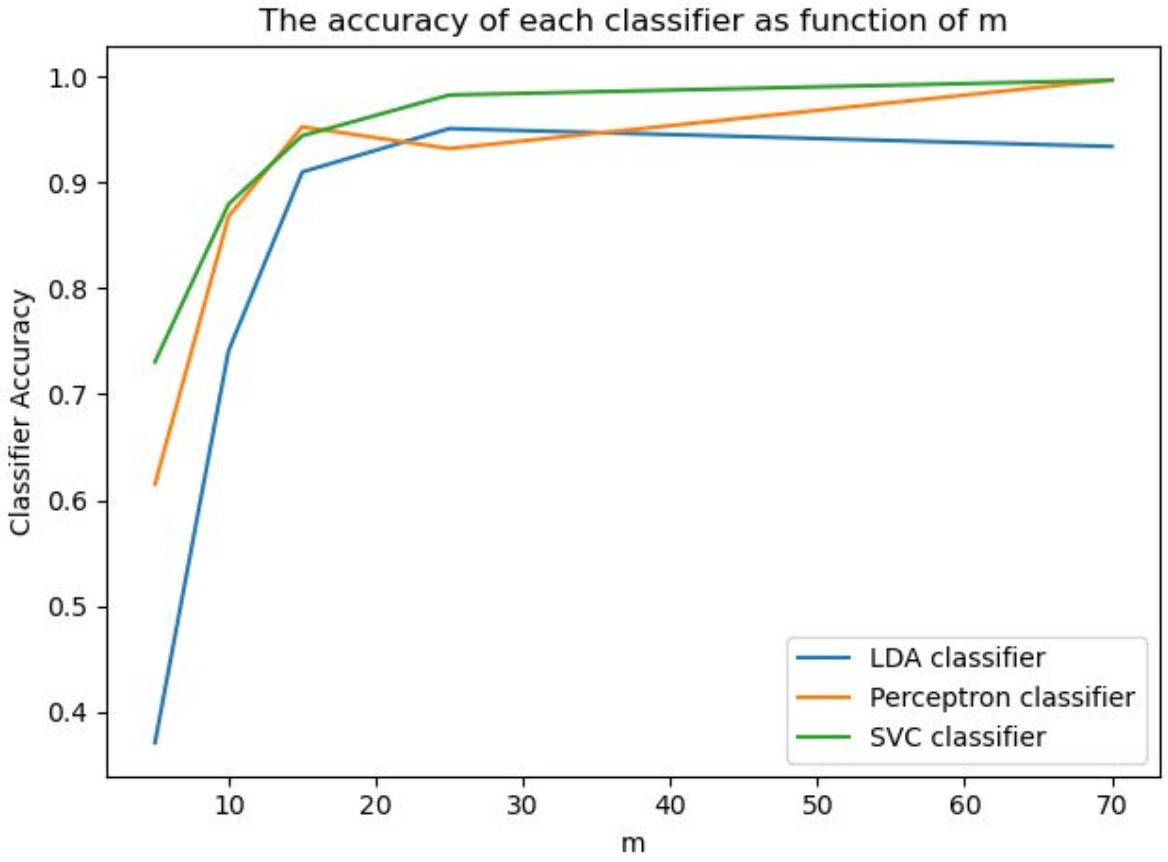

Drawn data points and hypothesis for m = 10
True hypothesis
Perceptron hypothesis
SVC hypothesis
Y = -1
Y = 1
x1
x2


Drawn data points and hypothesis for m = 5
True hypothesis
Perceptron hypothesis
SVC hypothesis
Y = -1
Y = 1
x1
x2
4
3
2
1
0
−1
−2
−3
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

10. We'll now add a test set to compare three of the above algorithms. We'll use the following procedure:

    - Draw training points $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ from the distribution $\mathcal{D}$ above and classify them according to a true hypothesis f (i.e. with labels $y_1, \ldots, y_m$). **Note:** The training data should always have points from two classes. So if you draw a training set where no point has $y_i = 1$ or no point has $y_i = -1$ then just draw a new dataset instead, until you get points from both types.
    - Draw k test points $\{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$ from the same distribution $\mathcal{D}$ and calculate their true labels as well.
    - Train a **Perceptron** classifier, an **SVM** classifier and an **LDA** classifier on $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$.
    - Calculate the accuracy of these classifiers (the fraction of test points that is classified correctly) on $\{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$.

    For each $m \in \{5, 10, 15, 25, 70\}$, repeat the above procedure 500 times with $k = 10000$ and save the accuracies (or just keep the mean accuracy, remember that the accuracy is a number between 0 to 1) of each classifier. Finally, plot the mean accuracy as function of m for each of the algorithms (SVM, Perceptron and LDA). Do not forget to add a legend to your graph. Add the plot to your PDF file.



11. Which classifier did better? why do you think that happened? No need for a formal argument, just explain what are the properties of the classifiers that cause these results.

המסווג מסוג svm נתן את התוצאות הטובות ביותר מבחינת דיוק.
כאשר באופן כללי ה-lda classifier נותן תוצאות פחות טובות.
נשים לב כי עבור 70 דגימות ה-perceptron classifier וה-svm classifier מגיעים לדיוק גבוה מאוד.
אך עבור 25 דגימות ה-svm classifier הגיע לדיוק טוב יותר.
נראה כי עבור מספר מספיק גדול של דגימות התוצאות של שני המסווגים קרובות, עבור מספר קטן יותר המsvm מראה יכולות טובות יותר.

הסבר לגבי הביצועים של svm לעומת lda:

LDA: Assumes: data is Normally distributed. All groups are identically distributed. LDA is the best discriminator available in case all assumptions are actually met. QDA, by the way, is a non-linear classifier.

SVM: Generalizes the Optimally Separating Hyperplane. This hyperplane assumes that all groups are totally separable. SVM makes no assumptions about the data at all, meaning it is a very flexible method.

SVM classification is an optimization problem, LDA has an analytical solution. The optimization problem for the **SVM has a dual and a primal formulation that allows the user to optimize over either the number of data points** or the number of variables, depending on which method is the most computationally feasible.

LDA makes use of the *entire* data set to estimate covariance matrices and thus is somewhat prone to outliers. SVM is optimized over a subset of the data, which is those data points that lie on the separating margin.

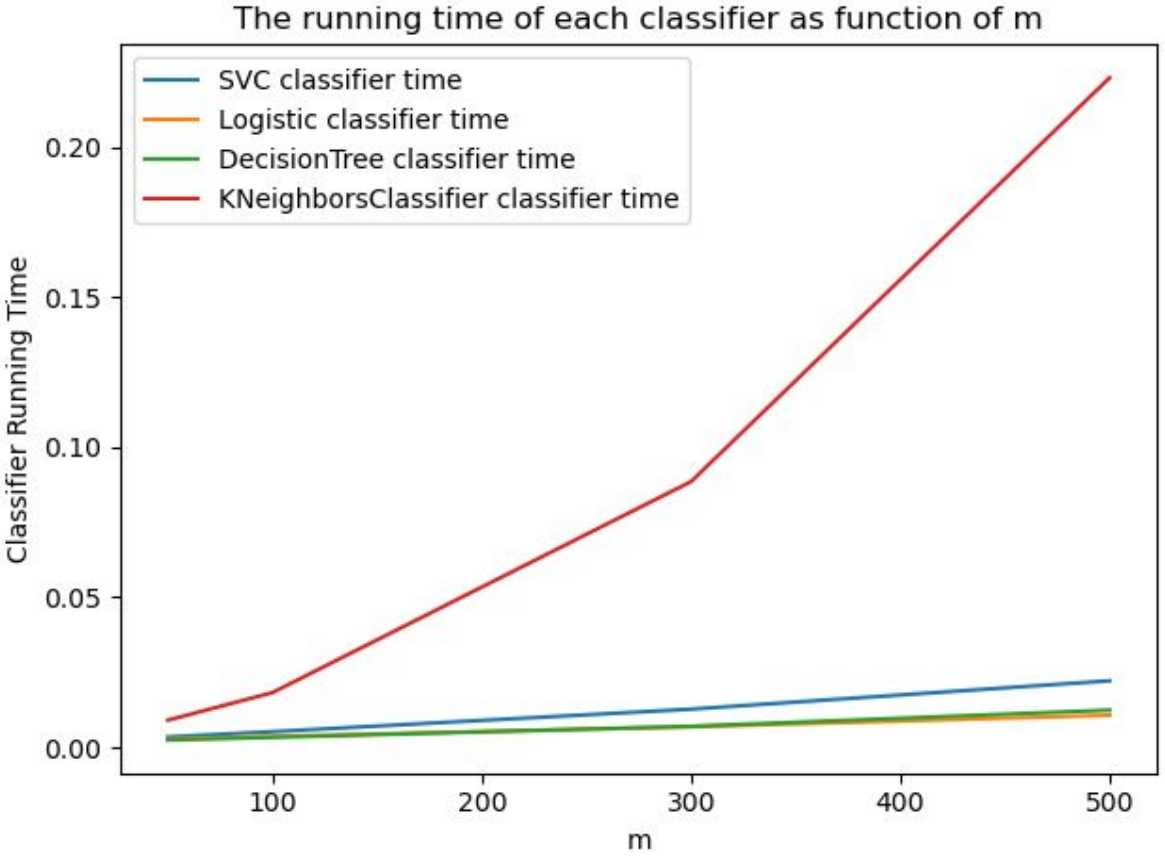
הסבר לגבי הביצועים של svm לעומת perceptron :

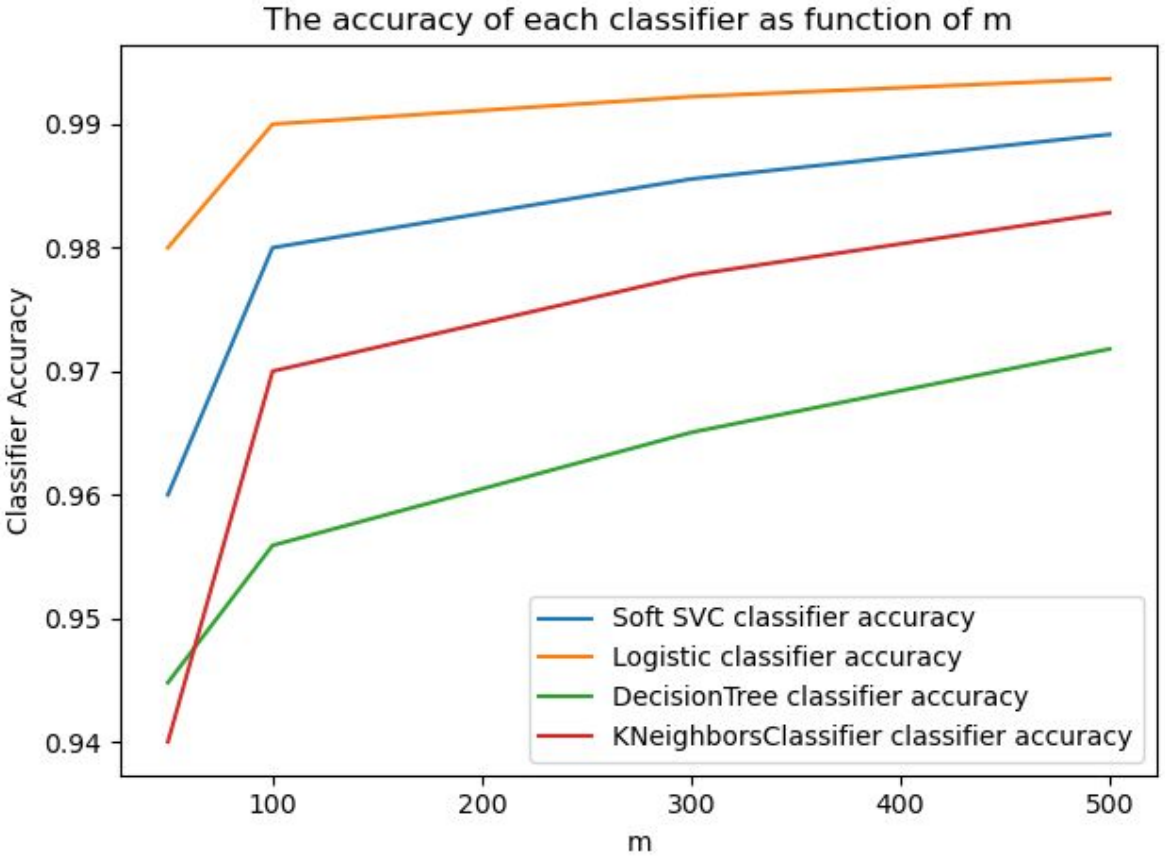
The major practical difference between a perceptron and SVM is that perceptrons can be trained online (i.e. their weights can be updated as new examples arrive one at a time) whereas SVMs cannot be. So, even though a SVM is usually a better classifier, perceptrons can still be useful because they are cheap and easy to re-train in a situation in which fresh training data is constantly arriving.

14. (similar to question 10, with different models) Consider the following procedure:

- Draw m training points $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ by choosing uniformly and at random from the train set. **Note:** The training data should always have points from two classes. So if you draw a training set where no point has $y_i = 1$ or no point has $y_i = -1$ then just draw a new dataset instead, until you get points from both types.
- Train a **logistic regression** classifier, an **Soft-SVM** classifier, a **decision tree** and a k-**nearest neighbors** classifier on $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$. You may choose any existing implementation of these algorithms as long as you note clearly in your solution which implementation you used. (Choose the regularization parameter for Soft-SVM and k for nearest neighbors to the best of your judgment - you're not expected to make the best possible choice at this stage. Although you are certainly encouraged to do your best and use side computations if you want.)
- Calculate their accuracy (the fraction of test points that is classified correctly) on the entire test set.

For each $m \in \{50, 100, 300, 500\}$, repeat the above procedure 50 times and save the elapsed running time and the accuracy (or just keep the mean accuracy, remember that the accuracy is a number between 0 to 1) of each classifier. Finally, plot the mean accuracy as function of m for each of the algorithms (SVM, Logistic regression, decision tree and nearest neighbors). Do not forget to add a legend to your graph. Add the plot to your PDF file. Discuss what you've seen about the difference in running time.


The running time of each classifier as function of m
SVC classifier time
Logistic classifier time
DecisionTree classifier time
KNeighborsClassifier classifier time
Classifier Running Time
0.00
0.05
0.10
0.15
0.20
100
200
300
400
500
m


The accuracy of each classifier as function of m
Soft SVC classifier accuracy
Logistic classifier accuracy
DecisionTree classifier accuracy
KNeighborsClassifier classifier accuracy
Classifier Accuracy
0.94
0.95
0.96
0.97
0.98
0.99
100
200
300
400
500
m

בגרף בו אנו מציגים את הדיוק של כל אחד מהמסווגים כפונקציה של מספר הדגימות נוכל להבחין כי:
ה - logistic classifier נותן את הדיוק הטוב ביותר (ללא השפעה של מספר הדגימות) ביחס לשאר המסווגים.
אחריו ה - soft svm classifier בהפרש מצומצם של 0.01 לכל היותר החל מ-100 דגימות ו0.02 עבור 50 דגימות.
עבור 50 דגימות decision tree classifier נותן דיוק טוב יותר לעומת k-nearest neighbors, אך החל מ 100 דגימות המצב מתהפך ו- k-nearest neighbors נותן דיוק טוב יותר לעומת decision tree classifier.

בגרף בו אנו מציגים את זמני הריצה נבחין כי kkn הינו בעל זמן הריצה הגבוה ביותר ביחס לאחרים וכמו כן ככל שמספר הדגימות גדל כך זמן הריצה גדל בצורה משמעותית.
עבור המסווגים האחרים כמות הדגימות משפיעה פחות על זמן הריצה לעומת kkn.
נבחין כי בkk בהנתן דגימה נמצא את k השכנים הקרובים ביותר ונחזיר תיוג על פי הצבעת הרוב.
כך שככל שיש יותר דגימות הבדיקה של השכנים הקרובים תגזול זמן רב יותר.

השוואה:

Logistic regression vs SVM :

- SVM can handle non-linear solutions whereas logistic regression can only handle linear solutions.
- Linear SVM handles outliers better, as it derives maximum margin solution.
- Hinge loss in SVM outperforms log loss in LR.

Logistic Regression vs Decision Tree :

- Decision tree handles colinearity better than LR.
- Decision trees cannot derive the significance of features, but LR can.
- Decision trees are better for categorical values than LR.

Logistic Regression vs KNN :

- KNN is a non-parametric model, where LR is a parametric model.
- KNN is comparatively slower than Logistic Regression.
- KNN supports non-linear solutions where LR supports only linear solutions.
- LR can derive confidence level (about its prediction), whereas KNN can only output the labels.

K Nearest Neighbor (KNN) finds the “nearest examples” in the training data and chooses the label associated with the nearest examples.The algorithm takes a long time to compute. K will dominate by size of the class.

Logistic regression and support vector machines are closely linked. Both can be viewed as taking a probabilistic model and minimizing some cost associated with misclassification based on the likelihood ratio.
Logistic regression focuses on maximizing the probability of the data. The farther the data lies from the separating hyperplane (on the correct side), the happier LR is.
An SVM tries to find the separating hyperplane that maximizes the distance of the closest points to the margin (the support vectors).

Soft-SVM

- If training sample is not linearly separable, Hard-SVM returns no solution
- We no longer assume linearly separable. **Let's relax** the constraint to yield soft-SVM

$$\underset{\mathbf{w},\boldsymbol{\xi}}{\operatorname{argmin}} \left(\lambda \|\mathbf{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i \right)$$

$$\text{s.t. } \forall i,\ y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

Summary - Soft SVM

- **Hypothesis class** $\mathcal{H}$: Half-spaces
- **Learning principle** (choosing $f \in \mathcal{H}$): Linear separator with Maximum Margin
- **Computational implementation**: Quadratic programming

Summary: Logistic regression

- **Hypothesis class**:

$$\mathcal{H}^d_{logistic} = \{\mathbf{x} \mapsto \pi(\langle \mathbf{x}, \mathbf{w} \rangle)\} .$$

- **Learning principle**: Maximum Likelihood
- **Computational implementation**: Convex optimization to maximize log-likelihood. Specialized iterative method / general convex solver

- So the maximum likelihood principle tells us to choose $h \in \mathcal{H}^d_{logistic}$, (equivalently, the vector $\mathbf{w} \in \mathbb{R}^{d+1}$) by finding

$$\hat{\mathbf{w}} := \underset{\mathbf{w} \in \mathbb{R}^{d+1}}{\operatorname{argmax}} \sum_{i=1}^{m} \left[y_i \langle \mathbf{x_i}, \mathbf{w} \rangle - \log \left(1 + e^{\langle \mathbf{x_i}, \mathbf{w} \rangle} \right) \right] .$$

LR gives calibrated probabilities that can be interpreted as confidence in a decision. LR gives us an unconstrained, smooth objective. LR can be used within Bayesian models. SVMs don't penalize examples for which the correct decision is made with sufficient confidence. This may be good for generalization.

Logistic Regression and trees differ in the way that they generate *decision boundaries* i.e. the lines that are drawn to separate different classes.
Decision Trees bisect the space into smaller and smaller regions, whereas Logistic Regression fits a single line to divide the space exactly into two. A single linear boundary can sometimes be limiting for Logistic Regression. if two classes are separated by a decidedly non-linear boundary, trees can better capture the division, leading to superior classification performance. However, when classes are not well-separated, trees are susceptible to overfitting the training data, so that Logistic Regression's simple linear boundary generalizes better.