IML EX3

Eliyahu Strugo 305589269|

May 14, 2020

1.Solution:

Let $t \in \{-1,1\}$.

$$Pr(y=t) \geq Pr(y=-t) = 1 - Pr(y=t) \Longleftrightarrow Pr(y=t) \geq \frac{1}{2}$$

And:

$$\underset{y\in\{-1,1\}}{argmax}\ Pr(x|y)\,Pr(y) = \underset{y\in\{-1,1\}}{argmax}\ \frac{Pr(x|y)\,Pr(y)}{Pr(x)} \underset{Bayes}{=} \underset{y\in\{-1,1\}}{argmax}\ Pr(y|x)$$

Hance:

$$h_{\mathcal{D}} = 1 \Longleftrightarrow Pr(y=1) \geq \frac{1}{2} \Longleftrightarrow Pr(y=1) \geq Pr(y=-1) \Longleftrightarrow 1 = \underset{y\in\{-1,1\}}{argmax}\ Pr(x|y)\,Pr(y)$$

$$h_{\mathcal{D}} = -1 \Longleftrightarrow Pr(y=1) < \frac{1}{2} \Longleftrightarrow Pr(y=-1) > Pr(y=1) \Longleftrightarrow -1 = \underset{y\in\{-1,1\}}{argmax}\ Pr(x|y)\,Pr(y)$$

Therefore , $h_{\mathcal{D}} = \underset{y\in\{-1,1\}}{argmax}\ Pr(x|y)\,Pr(y)$.

2.Solution:

$$\underset{y}{max}\ \{f(y|x)\} = \underset{y}{max}\ \{f(x|y)\,f(y)\} = \underset{y}{max}\,\{ln\,f(x|y) + ln\,f(y)\}$$

And:

$$\underset{y}{max}\,\{ln\,f(x|y) + ln\,f(y)\} = \underset{y}{max}\left\{-\frac{1}{2}(x-\mu_y)^T\Sigma^{-1}(x-\mu_y) - \frac{1}{2}ln\ (2\pi)^d \cdot det(\Sigma) + ln\,f(y)\right\}$$

$$= \underset{y}{max}\left\{-\frac{1}{2}(x-\mu_y)^T\Sigma^{-1}(x-\mu_y) + ln\,f(y)\right\}$$

Furthermore :

$$-\frac{1}{2}(x-\mu_y)^T\Sigma^{-1}(x-\mu_y) = -\frac{1}{2}\left(x^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu_y - \mu_y^T\Sigma^{-1}x + \mu_y^T\Sigma^{-1}\mu_y\right)$$

Since $c = \mu_y^T\Sigma^{-1}x$ is a scalar then $c^T = x^T\Sigma^{-1}\mu_y = c$ so we get:

$$-\frac{1}{2}(x-\mu_y)^T\Sigma^{-1}(x-\mu_y) = -\frac{1}{2}\left(x^T\Sigma^{-1}x - 2x^T\Sigma^{-1}\mu_y + \mu_y^T\Sigma^{-1}\mu_y\right) = -\frac{1}{2}x^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu_y - \frac{1}{2}\mu_y^T\Sigma^{-1}\mu_y$$

Thus:

$$\underset{y}{max}\ \{f(y|x)\} = \underset{y}{max}\left\{-\frac{1}{2}(x-\mu_y)^T\Sigma^{-1}(x-\mu_y) + ln\,f(y)\right\} = \underset{y}{max}\left\{-\frac{1}{2}x^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu_y - \frac{1}{2}\mu_y^T\Sigma^{-1}\mu_y + ln\,f(y)\right\}$$

$$= \underset{y}{max}\left\{x^T\Sigma^{-1}\mu_y - \frac{1}{2}\mu_y^T\Sigma^{-1}\mu_y + ln\,f(y)\right\} = \underset{y}{max}\,\{\delta_y\} = h_{\mathcal{D}}$$

3.Solution:

Let $N_k = |\{(x,y) \in S_n \,|y = k\}|$ s.t $k \in \{-1,1\}$. Since for every $i \in [n]$ we know to which class x_i belongs then we can use the distributions $p(x^n, y^n)$in order to estimate the parameters.

For the prior and the means I will use MLE estimator:

$$log\, p(x^n, y^n) = log \prod_{i=1}^{n} p(x_i, y_i) = \sum_{i=1}^{n} log\, p(x_i, y_i) = \sum_{i=1}^{n} log\, p(x_i|y_i)\, p(y_i) = \sum_{i=1}^{n} log\, p(x_i|y_i)\, \pi_{y_i} = \sum_{i=1}^{n} log\, p(x_i|y_i) + log\, \pi_{y_i}.$$

By using Lagrange Multipliers - the constraint is $\pi_1 + \pi_{-1} = 1$:

$$\frac{\partial\left(log\, p(x^n, y^n) - \lambda(\pi_1 + \pi_{-1} - 1)\right)}{\partial \pi_k} = \sum_{i=1}^{n} \frac{\partial log\, \pi_{y_i}}{\partial \pi_k} = \sum_{y_i = k} \frac{1}{\pi_{y_i}} - \lambda = 0$$

$$\Rightarrow \pi_k = \frac{N_k}{\lambda} \Rightarrow \pi_1 + \pi_{-1} = 1 = \frac{N_1 + N_{-1}}{\lambda} = \frac{n}{\lambda} \Rightarrow \lambda = n$$

Hence $\hat{\pi}_k = \frac{N_k}{n}$.

Now estimating μ_k :

$$\frac{\partial\left(log\, p(x^n, y^n)\right)}{\partial \mu_k} = \sum_{i=1}^{n} \frac{\partial log\, p(x_i|y_i)}{\partial \mu_k} = \sum_{i=1}^{n} \frac{\partial - \frac{1}{2}(x - \mu_k)^T \Sigma^{-1}(x - \mu_k)}{\partial \mu_k} = \sum_{i:y_i = k} -\frac{1}{2}\Sigma^{-1}(x_i - \mu_k)$$

Therefore,

$$\sum_{i:y_i = k} -\frac{1}{2}\Sigma^{-1}(x_i - \mu_k) = 0 \Rightarrow \sum_{i:y_i = k} \Sigma^{-1} x_i = N_k \Sigma^{-1} \mu_k \Rightarrow \mu_k = \frac{1}{N_k} \sum_{i:y_i = k} x_i$$

Hence $\hat{\mu}_k = \frac{1}{N_k} \sum\limits_{i:y_i = k} x_i$.

Now we have to estimate Σ. Since both distributions share the same Σ we use "pooled covariance" to consider them both. Let $\hat{\Sigma}_k = \frac{1}{N_k - 1} \sum\limits_{i:y_i = k} (x_i - \mu_k)(x_i - \mu_k)^T$ where $k \in \{-1,1\}$. This is the estimator we saw in the class.

So by using "Pooled Covariance" we get :

$$\hat{\Sigma} = \frac{\sum\limits_{i:y_i = 1} (x_i - \mu_k)(x_i - \mu_k)^T + \sum\limits_{i:y_i = -1} (x_i - \mu_k)(x_i - \mu_k)^T}{N_1 + N_{-1} - 2} = \frac{1}{n-2} \sum_{k \in \{\pm 1\}} \sum_{i:y_i = k} (x_i - \mu_k)(x_i - \mu_k)^T$$

Thus, $\hat{\Sigma} = \frac{1}{n-2} \sum\limits_{k \in \{\pm 1\}} \sum\limits_{i:y_i = k} (x_i - \mu_k)(x_i - \mu_k)^T$

4.Solution:

Let $\delta : X \to \{-1,1\}$ be a classifier for the Spam filter where $Spam \equiv 1$ and $Not - Spam \equiv -1$.

The classifier can have two possible misclassification:

- $\delta(x) = 1$ while x is Not-Spam
- $\delta(x) = -1$ while x is Spam

Let $D_0 = \{x \in X \,|\, \delta(x) = 1\}$ and $D_1 = \{x \in X \,|\, \delta(x) = -1\}$

- Type I : false positive error is $\alpha = p\left(x \in D_1 | y = 1\right)$. The probability of classifying an email x as Not-Spam while the email is Spam.
- Type II : false negative error is $\beta = p\left(x \in D_0 | y = -1\right)$. The probability of classifying an email x as Spam while the email is Not-Spam.

It's open to interpretation but usally missing an important email because it's been classified as a spam ($Type\, I$) could cause more damge then read a spam email ($Type\, II$) so we would care more about the value ofα .

5.Solution:

Let $Q = 2 \cdot \begin{bmatrix} I_{n-1} & 0 \\ 0 & 0 \end{bmatrix}$, $A = (-1) \cdot \begin{bmatrix} y_1 x_{1,1} & \dots & y_1 x_{1,n-1} & 1 \\ y_2 x_{2,1} & \dots & y_2 x_{2,n-1} & 1 \\ \vdots & & & \vdots \\ y_m x_{m,1} & \dots & y_m x_{m,n-1} & 1 \end{bmatrix}$, v= $\begin{bmatrix} w_1 \\ \vdots \\ w_{n-1} \\ b \end{bmatrix}$, d= $\begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}$ and $a = 0_n$.

Then, $\frac{1}{2} \cdot v^T Q v = w^T w = \|w\|^2$ and $\forall i, y_i \langle w, x_i \rangle \geq 1 \Leftrightarrow -Av \geq 1 \Leftrightarrow Av \leq 1$.

Why it's interesting:

Since the objective function is convex and all the constraints are linear, we can solve this problem efficiently using standard quadratic programming (QP) by using convex optimization techniques .

6.Solution:

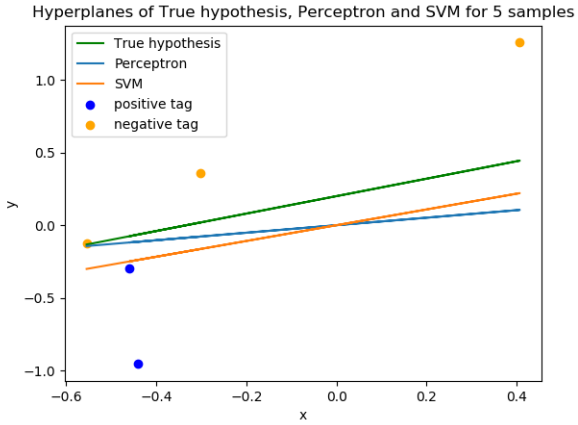
Let ξ_i for some $i \in [n]$. if $\xi_i \geq 1 - y_i \langle w, x_i \rangle$ and $\xi_i \geq 0$ then:

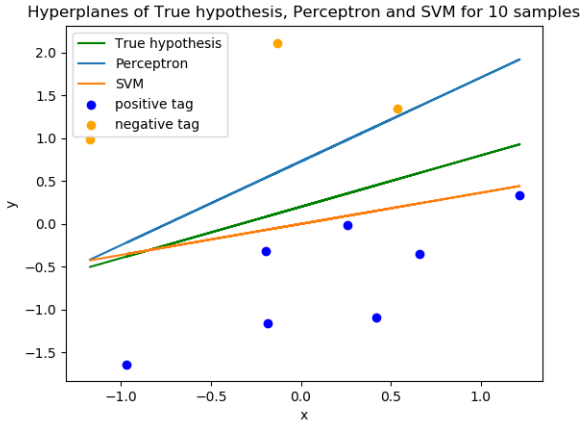
- if $0 > 1 - y_i \langle w, x_i \rangle$ then the minimum value that ξ_i can get is 0
- else $0 \leq 1 - y_i \langle w, x_i \rangle$ so the minimum value that ξ_i can get is $0 \leq 1 - y_i \langle w, x_i \rangle$

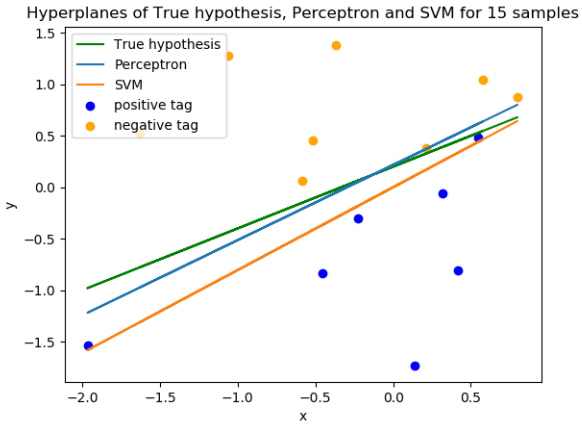
Hence , $\xi_i = min\left\{a | \, a \geq 1 - y_i \langle w, x_i \rangle \text{ and } a \geq 0 \text{ then}\right\} \Leftrightarrow \xi_i = max\left\{0, 1 - y_i \langle w, x_i \rangle\right\}$.Therefore:

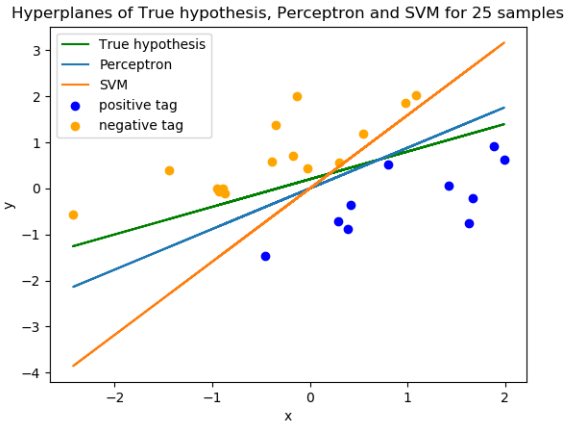
$$\min_{w, \{\xi_i\}} \left\{ \frac{\lambda}{2} \cdot \|w\|^2 + \frac{1}{m} \sum_i \xi_i \, | \, \forall i, \xi_i \geq 1 - y_i \langle w, x_i \rangle \; and \, \xi_i \geq 0 \right\} = \min_w \left\{ \frac{\lambda}{2} \cdot \|w\|^2 + \frac{1}{m} \sum_i l^{hinge} \left(y_i \langle w, x_i \rangle \right) \right\}$$

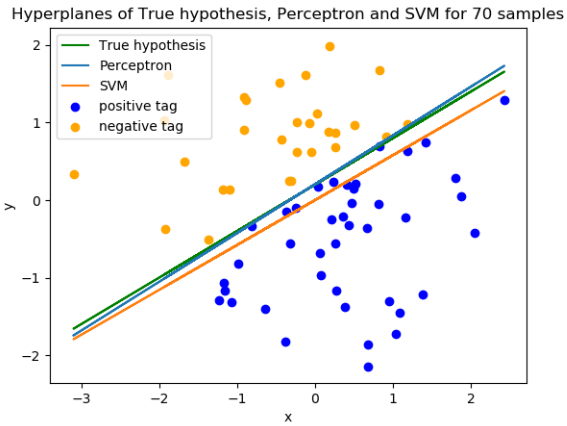
9.Solution:



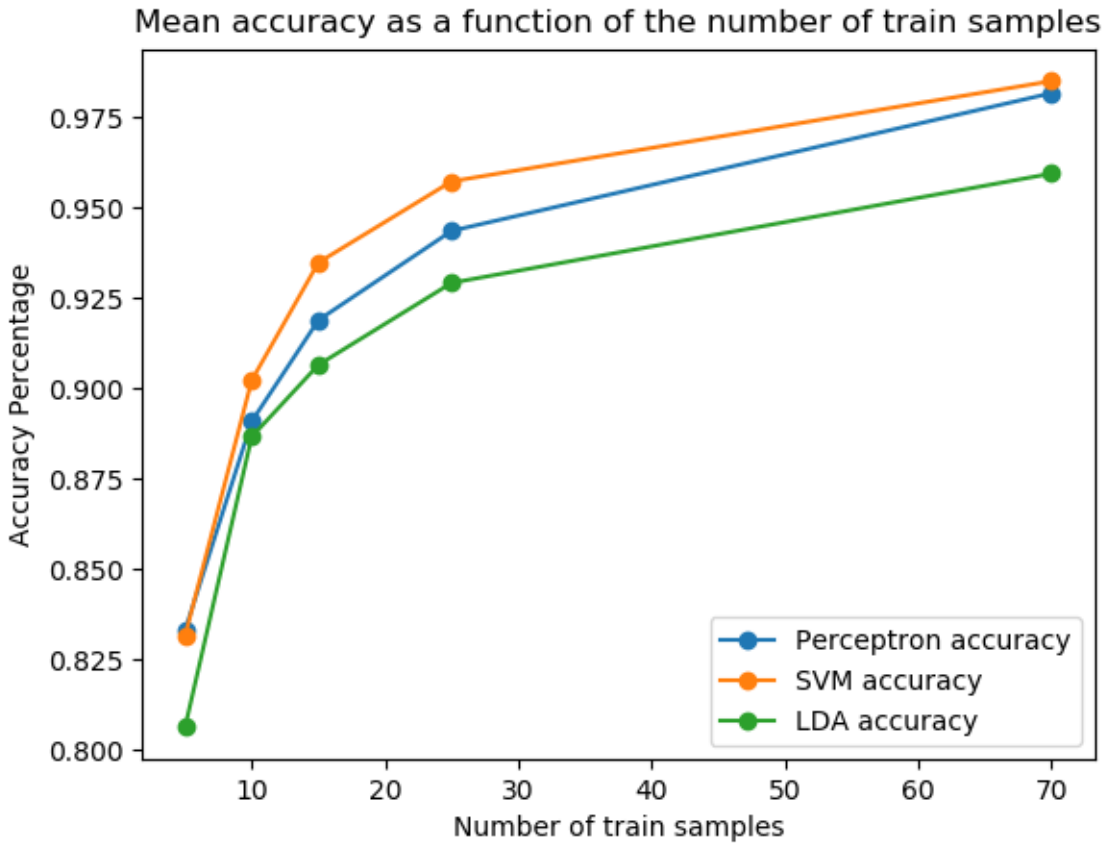




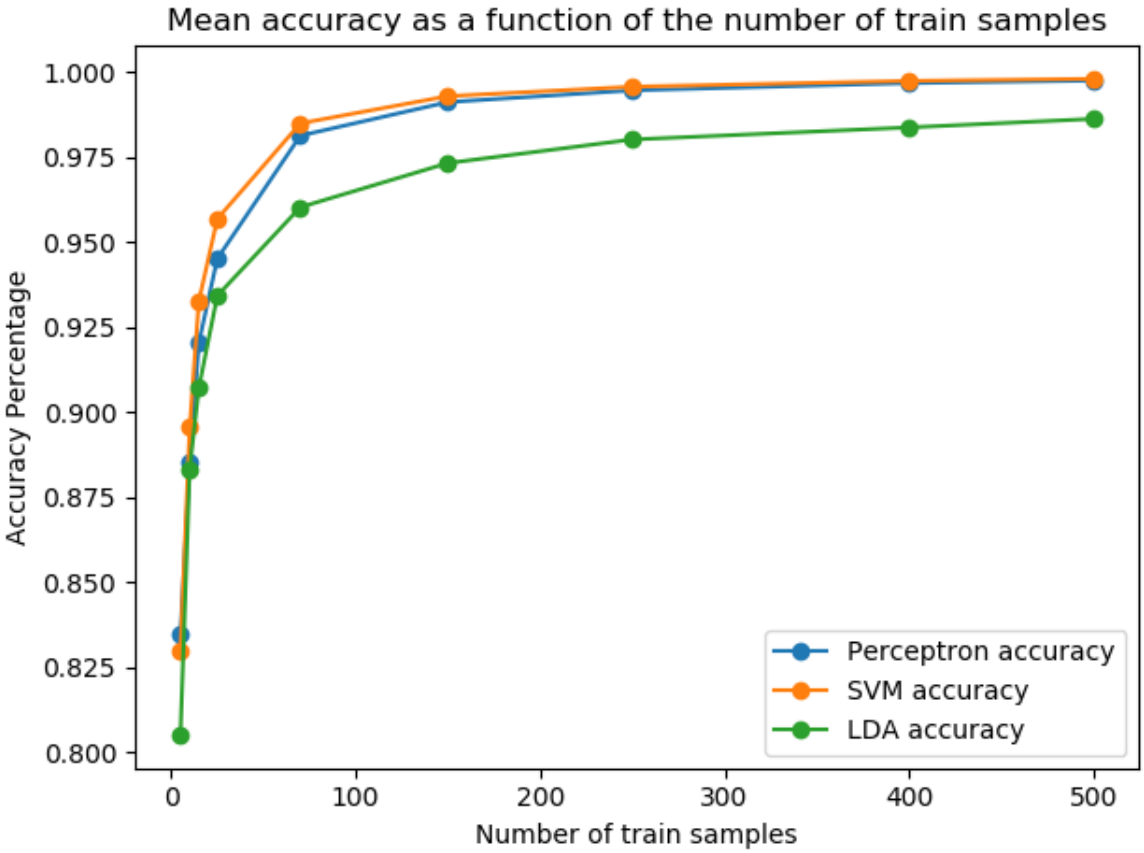




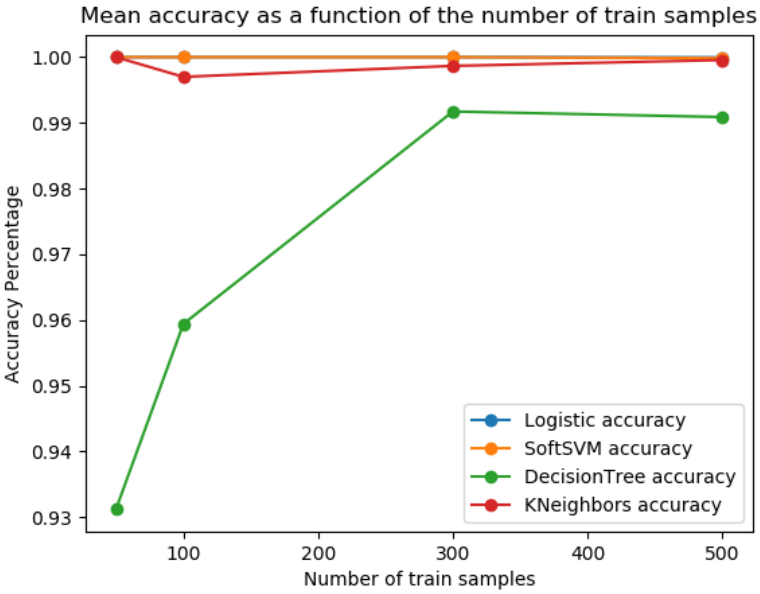
10-11.Solution:

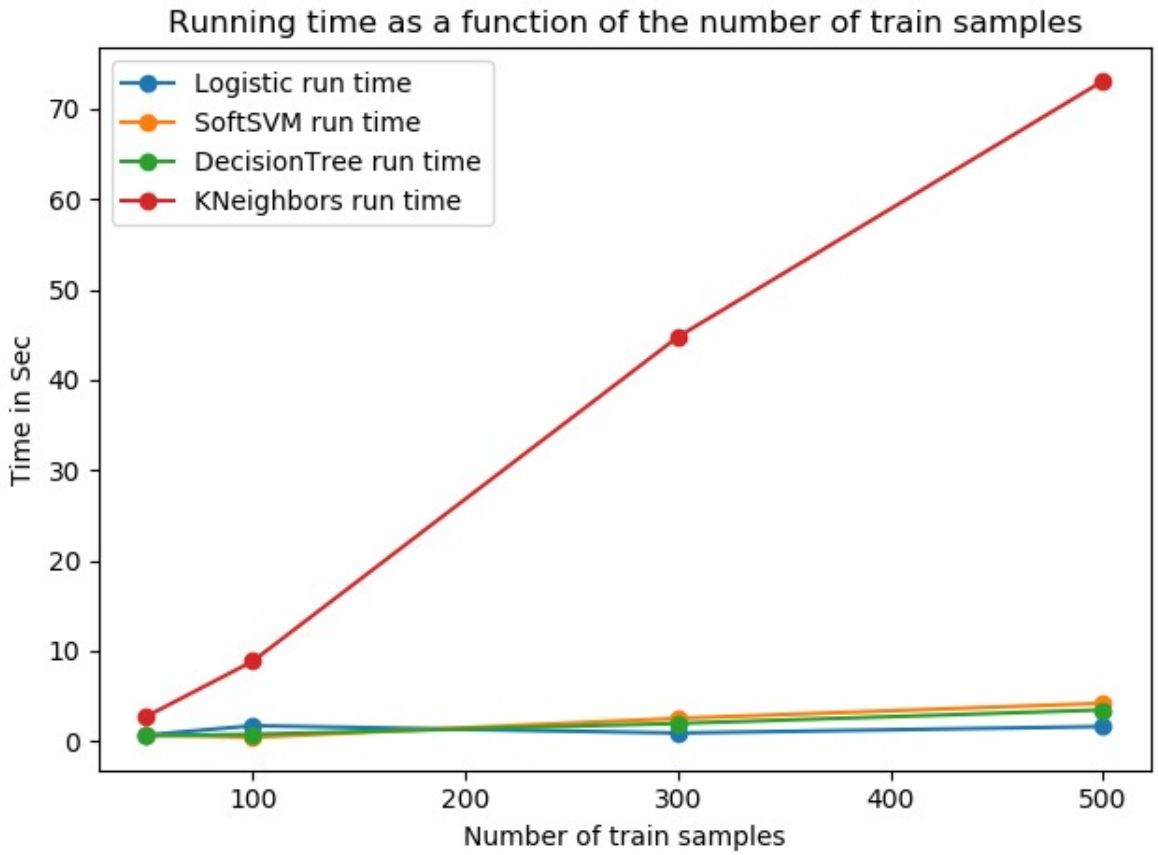


The best classifier in iterms of accuracy here is SVM. An explaintion for the heigher prefomance of SVM compare to Perceptron could be that the Perceptron classifier does not necessarily convarges to an hyperplane that has a maximum margin. SVM on ther other hand finds an hyperplane with the widest margin and by that it lowers the chances that new points would me misclassificated. The LDA seems to have the poorest prefomance, which is interesting - since LDA is optimal in Bayesian view, but we do not know the parameters and we had to estimate them. For example I used MLE estimatior for the prior which can lead to some misclassification. Since as the sample grows the term $log\, p\,(y)$ has little effect on the posterior then we could expect that increasing in the number of train samples will improve the accuracy of LDA :


Mean accuracy as a function of the number of train samples
Accuracy Percentage
Number of train samples
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
0
100
200
300
400
500
Perceptron accuracy
SVM accuracy
LDA accuracy

14.Solution:


Mean accuracy as a function of the number of train samples
Accuracy Percentage
Number of train samples
1.00
0.99
0.98
0.97
0.96
0.95
0.94
0.93
100
200
300
400
500
Logistic accuracy
SoftSVM accuracy
DecisionTree accuracy
KNeighbors accuracy



	$m = 50$	$m = 100$	$m = 300$	$m = 500$
Logistic	1	1	1	0.99996
SVM	1	1	1	0.9997
KNN	0.9992	0.997	0.9983	0.9992
DT	0.9372	0.964	0.992	0.9914

It seems that KNN has mush longer running time for this problem where I used sklearn algorithem and $k = 1$. The main issue with the algorithm that it computes the distance between the images using a Euclidean distance: we regarded each 28*28 pixel image as a point in 784-space, and computes the distance between two such points for every pair of images which could take a lot of time (as it takes for 300 and 500 samples).