Introduction To Machine Learning: EX3

Or Tal 305793168

Bayes Optimal and LDA

[Relevant material - Lecture 3]

Consider binary classification with sample space $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{\pm 1\}$. One way to model the data generation process is to assume that our samples are drawn i.i.d from an unknown **joint** distribution $\mathcal{D}$ over $\mathcal{X} \times \{\pm 1\}$. (Namely, we draw the sample and the label together from a joint distribution over $\mathcal{X} \times \{\pm 1\}$.) In this question you'll learn about two concepts that were not discussed in the lecture: The Bayes Optimal Classifier, and Linear Discriminent Analysis (LDA).

1. If we knew $\mathcal{D}$, our best predictor would have been assigning the class with the higher probability:

$$\forall \mathbf{x} \in \mathcal{X}\ h_{\mathcal{D}}(\mathbf{x}) = \begin{cases} +1 & \Pr(y = 1|\mathbf{x}) \geqslant \frac{1}{2} \\ -1 & otherwise \end{cases}$$

where the probability is over $\mathcal{D}$. This classifier is known as the **Bayes Optimal** classifier.

Show that

$$h_{\mathcal{D}} = \underset{y \in \{\pm 1\}}{\operatorname{argmax}} \Pr(\mathbf{x}|y) \Pr(y).$$

Using Bayes Rule we may see that: $\mathbb{P}(y = Y|x = X) = \frac{\mathbb{P}(x = X|y = Y)\mathbb{P}(y=Y)}{\mathbb{P}(x=X)}$

Hence for any $\boldsymbol{x_0} \in \mathcal{X}$: $h_{\mathcal{D}}(\boldsymbol{x_0}) = \begin{cases} 1, & \mathbb{P}(y = 1|\boldsymbol{x} = \boldsymbol{x_0}) = \frac{\mathbb{P}(\boldsymbol{x} = \boldsymbol{x_0}|y = 1)\mathbb{P}(y=1)}{\mathbb{P}(\boldsymbol{x}=\boldsymbol{x_0})} \geq \frac{1}{2} \\ -1, & \mathbb{P}(y = -1|\boldsymbol{x} = \boldsymbol{x_0}) = \frac{\mathbb{P}(\boldsymbol{x} = \boldsymbol{x_0}|y = -1)\mathbb{P}(y=-1)}{\mathbb{P}(\boldsymbol{x}=\boldsymbol{x_0})} > \frac{1}{2} \end{cases}$

And as $\mathbb{P}(x = x_0)$ is the same at both terms and as we have only 2 options for y, we may conclude that the term $\frac{\mathbb{P}(\boldsymbol{x} = \boldsymbol{x_0}|y = 1)\mathbb{P}(y=1)}{\mathbb{P}(\boldsymbol{x}=\boldsymbol{x_0})} \geq \frac{1}{2}$, is equivalent with $\mathbb{P}(\boldsymbol{x} = \boldsymbol{x_0}|y = 1)\mathbb{P}(y = 1) \geq \mathbb{P}(\boldsymbol{x} = \boldsymbol{x_0}|y = -1)\mathbb{P}(y = -1)$

And therefore $h_{\mathcal{D}}(\boldsymbol{x_0}) = \begin{cases} 1, & \mathbb{P}(\boldsymbol{x} = \boldsymbol{x_0}|y = 1)\mathbb{P}(y = 1) \geq \mathbb{P}(\boldsymbol{x} = \boldsymbol{x_0}|y = -1)\mathbb{P}(y = -1) \\ -1, & \mathbb{P}(\boldsymbol{x} = \boldsymbol{x_0}|y = -1)\mathbb{P}(y = -1) > \mathbb{P}(\boldsymbol{x} = \boldsymbol{x_0}|y = 1)\mathbb{P}(y = 1) \end{cases}$

So we may conclude $\forall \boldsymbol{x_0} \in \mathcal{X}$: $h_{\mathcal{D}}(\boldsymbol{x_0}) = argmax_{\{y \in \pm 1\}}\mathbb{P}(\boldsymbol{x} = \boldsymbol{x_0}|y)\mathbb{P}(y) \Rightarrow as\ requested$ ■

2. Assume that $\mathcal{X} = \mathbb{R}^d$ and that $\mathbf{x}|y \sim \mathcal{N}(\mu_y, \Sigma)$ for some mean vector $\mu_y \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$ (that is, the covariance matrix Σ is the same for both $y \in \{\pm 1\}$, but the expectation μ_y is different for each $y \in \{\pm 1\}$). In other words,

$$f(\mathbf{x}|y) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_y)^\top \Sigma^{-1}(\mathbf{x}-\mu_y)\right\}$$

where f is the density function for the multivariate normal distribution. Show that in this case, if we knew μ_{+1}, μ_{-1} and Σ then the Bayes Optimal classifier is

$$h_{\mathcal{D}}(\mathbf{x}) = \operatorname*{argmax}_{y\in\{\pm 1\}} \delta_y(\mathbf{x}),$$

where δ_{+1} and δ_{-1} are functions $\mathbb{R}^d \to \mathbb{R}$ given by

$$\delta_y(\mathbf{x}) = \mathbf{x}^\top \Sigma^{-1}\mu_y - \frac{1}{2}\mu_y^\top \Sigma^{-1}\mu_y + \ln \Pr(y) \qquad y \in \{\pm 1\}$$

Dealing with continuous variables, the density function is an estimation $\mathbb{P}(x_0 < x \leq x_0 + \epsilon)$ for some small epsilon choice, hence we may think of this as an approximation of the discrete probability.

By understanding this conclusion, using the general conclusion from previous section we may conclude that in this case

$$h_{\mathcal{D}}(\boldsymbol{x}) = argmax_{\{y\in\pm 1\}} f(\boldsymbol{x}|y)\mathbb{P}(y) = argmax_{\{y\in\pm 1\}} \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \mathrm{e}^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_y})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu_y})} \mathbb{P}(y)$$

And as $\frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}}$ is a finite constant (Σ is invertible $\Rightarrow \det\Sigma \neq 0$) , and $e^\alpha, \mathbb{P}(y) > 0$

Hence maximizing the above is equivalent to maximizing the log of the expression.

We could ignore the constant as id contributes equally to every case ang get:

$$h_{\mathcal{D}}(\boldsymbol{x}) = argmax_{\{y\in\pm 1\}} f(\boldsymbol{x}|y)\mathbb{P}(y) = argmax_{\{y\in\pm 1\}} \ln\left(e^{-\frac{1}{2}(\boldsymbol{x}-\mu_y)^T \Sigma^{-1}(\boldsymbol{x}-\mu_y)} \mathbb{P}(y)\right)$$
$$= argmax_{\{y\in\pm 1\}} \left(-\frac{1}{2}(\boldsymbol{x}-\mu_y)^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu_y}) + \ln \mathbb{P}(y)\right)$$

We know the Σ is a positive semi-definite matrix, therefore Σ^{-1} would also be symmetric

$$(\boldsymbol{x}-\boldsymbol{\mu_y})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu_y}) = (\Sigma^{-1}\boldsymbol{x} - \Sigma^{-1}\boldsymbol{\mu_y})^T(\boldsymbol{x}-\boldsymbol{\mu_y}) = \boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x} - \boldsymbol{\mu_y}^T\Sigma^{-1}\boldsymbol{x} - \boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu_y} + \boldsymbol{\mu_y}^T\Sigma^{-1}\boldsymbol{\mu_y}$$

Note that $\boldsymbol{\mu_y}^T\Sigma^{-1}\boldsymbol{x}$ dimensions are $(1,d)\times(d,d)\times(d,1) = (1\times 1) \Rightarrow$ hence it is a constant so we may conclude:

$$\boldsymbol{\mu_y}^T\Sigma^{-1}\boldsymbol{x} = const = (\boldsymbol{\mu_y}^T\Sigma^{-1}\boldsymbol{x})^T = \boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu_y}$$

Furthermore, we may see that $\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x}$ is not dependent on $\boldsymbol{\mu_y}$,so for $\boldsymbol{\mu_1}, \boldsymbol{\mu_{-1}}$ it should have the same value.

We may then conclude that :

$$argmax_{\{y\in\pm 1\}}\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_y})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu_y}) + \ln\mathbb{P}(y)\right) = argmax_{\{y\in\pm 1\}}\left(-\frac{1}{2}(\boldsymbol{\mu_y}^T\Sigma^{-1}\boldsymbol{\mu_y} - 2\cdot\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu_y}) + \ln\mathbb{P}(y)\right)$$
$$= argmax_{\{y\in\pm 1\}}\left(\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu_y} - \frac{1}{2}\boldsymbol{\mu_y}^T\Sigma^{-1}\boldsymbol{\mu_y} + \ln\mathbb{P}(y)\right) = argmax_{\{y\in\pm 1\}}\delta_y(\boldsymbol{x}) \Rightarrow as\ requested\ \blacksquare$$

3. In practice, we don't know μ_{+1}, μ_{-1}, Σ and $\Pr(y)$. In order to turn the above into a classifier, given a training set $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$, we need to estimate them. Write your formula for estimating μ_{+1}, μ_{-1}, Σ and $\Pr(y)$ based on S.

We know that $(\boldsymbol{x}|y) \sim N(\mu_y, \Sigma)$, meaning that samples are drawn from two identical gaussians with different mean.

Giving some intuition, in the 1D case:

the probability that $\boldsymbol{x}$ was taken from the red sample would in general be $\frac{num\ of\ red\ samples\ in\ training\ set}{num\ of\ overall\ training\ samples}$ considering iid choice of $\boldsymbol{x}$.

In order to estimate μ_1, μ_{-1} we would take the $\boldsymbol{x}\ axis$ average of all red and blue labels accordingly

Calculating the variance of $\boldsymbol{x}$ is the same taking the sum of variances per class divided by (# samples – # classes).

Expanding to $\boldsymbol{d}$ dimnsional space:

$\mathbb{P}(y)$ would still be computed from $\frac{num\ of\ 'y'\ samples\ in\ training\ set}{num\ of\ overall\ training\ samples}$

As our sample space is finite, we would compute mean and covariance numerically.

$$\mu_y = \frac{1}{m} \cdot \sum_{i=1}^{m} \mathbb{I}(y_i = y)\, \boldsymbol{x_i}$$

For the estimation of Σ, we would like to estimate the common empirical covariance matrix as following:

Where K= num of classes = 2 $\Rightarrow \Sigma = \frac{1}{m-K} \sum_{k \in \{-1,1\}} \sum_{y_i = k} (\boldsymbol{x_i} - \mu_{y_i})(\boldsymbol{x_i} - \mu_{y_i})^T$

Spam

[Relevant material - Lecture 3]

4. You are building a spam filter - a classifier that receives an email and decides whether it's a spam message or not. What are the two kinds of errors that your classifier could make? Which of them is the error we really don't want to make? Which of the labels $\{\text{spam}, \text{not-spam}\}$ should be the **negative** label and which should be the **positive** label, if we want the false-positive error (Type-I error) to be the error we really don't want to make?

As seen in class, false positive should be the more "significant" error, therefore intuitively, having spam in the inbox would be less significant, than filtering a real message (say someone won the email-lottery and it was classified as spam)

So we define the following errors:

- False positive: classify real email as spam
- False negative: classify spam as relevant email

And accordingly:

- Negative label = spam
- Positive label = not-spam

5. The canonical form of a Quadratic Program (QP) is:

$$\underset{\mathbf{v}\in\mathbb{R}^n}{\operatorname{argmin}}\left(\frac{1}{2}\mathbf{v}^\top Q\mathbf{v}+\mathbf{a}^\top\mathbf{v}\right)$$
$$\text{s.t. } A\mathbf{v}\leqslant\mathbf{d},$$

where $Q\in\mathbb{R}^{n\times n}, A\in\mathbb{R}^{m\times n}, \mathbf{a}\in\mathbb{R}^n, \mathbf{d}\in\mathbb{R}^m$ are fixed vectors and matrices.

Write the Hard-SVM problem as a QP problem in canonical form. Specifically, using the Hard-SVM problem formulation

$$\underset{(\mathbf{w},b)}{\operatorname{argmin}}\,||\mathbf{w}||^2 \text{ s.t. } \forall i, y_i(\langle\mathbf{w},\mathbf{x}_i\rangle+b)\geqslant 1.$$

what are the values of $Q, A, \mathbf{a}, \mathbf{d}$ that express this problem as a QP in canonical form?

Minimizing the L_2 norm is equivalent to minimize the expression $\boldsymbol{w^Tw}$

$y_i(\langle \boldsymbol{w},\boldsymbol{x_i}\rangle+b)=y_i\boldsymbol{w^Tx_i}+y_ib \Rightarrow$ and as $\boldsymbol{w^Tx_i}$ is a scalar: $y_i(\langle \boldsymbol{w},\boldsymbol{x_i}\rangle+b)=y_i(\boldsymbol{x_i^Tw}+b)$

Say $\boldsymbol{w},\boldsymbol{x_i}\in\mathbb{R}^n$, hence stacking m equations that represent the above constraint will produce:

$$\begin{bmatrix} y_1\boldsymbol{x}_1^T \\ \vdots \\ y_m\boldsymbol{x}_m^T \end{bmatrix}\boldsymbol{w}\geq(1-b)\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}\Rightarrow\begin{bmatrix} -y_1\boldsymbol{x}_1^T \\ \vdots \\ -y_m\boldsymbol{x}_m^T \end{bmatrix}\boldsymbol{w}\leq(b-1)\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

So for $\left\{Q=2\cdot I_{n\times n},\ \boldsymbol{a}=\boldsymbol{0}_{n\times 1},\ A=\begin{bmatrix} -y_1\boldsymbol{x}_1^T \\ \vdots \\ -y_m\boldsymbol{x}_m^T \end{bmatrix}_{m\times n},\ \boldsymbol{d}=(b-1)\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix},\ \boldsymbol{v}=\boldsymbol{w}\right\}$ it's simple to see that:

$$argmin_{\{\boldsymbol{w}\in\mathbb{R}^n\}}\left|\left|\boldsymbol{w}\right|\right|^2=argmin_{\{\boldsymbol{w}\in\mathbb{R}^n\}}\left(\boldsymbol{w^Tw}\right)=argmin_{\{\boldsymbol{w}\in\mathbb{R}^n\}}\left(\frac{1}{2}\boldsymbol{w}^T2I\boldsymbol{w}+\boldsymbol{0}^T\boldsymbol{w}\right)=argmin_{\{\boldsymbol{v}\in\mathbb{R}^n\}}\left(\frac{1}{2}\boldsymbol{v}^TQ\boldsymbol{v}+\boldsymbol{a}^T\boldsymbol{v}\right)$$

$$\begin{bmatrix} -y_1\boldsymbol{x}_1^T \\ \vdots \\ -y_m\boldsymbol{x}_m^T \end{bmatrix}\boldsymbol{w}=A\boldsymbol{v}\leq(b-1)\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}=\boldsymbol{d}$$

We may conclude that under the above values, we may describe the wanted Hard-SVM problem as a QP problem in canonical form, as requested ■

6. In the Soft-SVM we defined the problem:

$$\arg\min_{\mathbf{w},\{\xi_i\}}\frac{\lambda}{2}||\mathbf{w}||^2+\frac{1}{m}\sum_{i=1}^m\xi_i \text{ s.t. } \forall_i, y_i\langle\mathbf{w},\mathbf{x}_i\rangle\geqslant 1-\xi_i \text{ and } \xi_i\geqslant 0$$

Show that this problem is equivalent to the problem (namely that these problem have the same solutions)

$$\arg\min_{\mathbf{w}}\frac{\lambda}{2}||\mathbf{w}||^2+\frac{1}{m}\sum_{i=1}^m\ell^{hinge}(y_i\langle\mathbf{w},\mathbf{x}_i\rangle),$$

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

It is given that $\forall i$: $y_i\langle\boldsymbol{w},\boldsymbol{x_i}\rangle\geq 1-\xi_i\Rightarrow\xi_i\geq 1-y_i\langle\boldsymbol{w},\boldsymbol{x_i}\rangle\ \wedge\ \xi_i\geq 0$

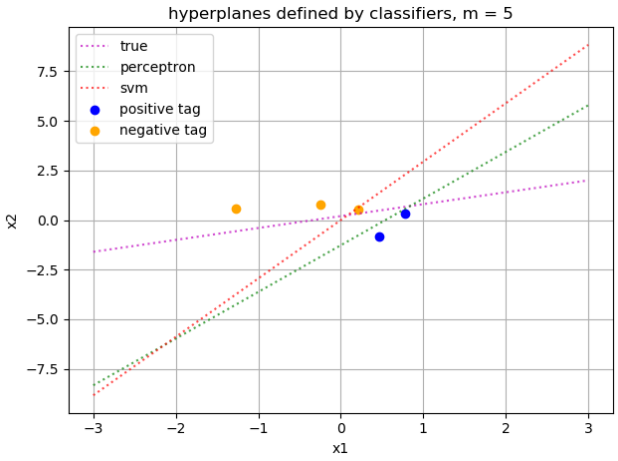
And so we may see that minimizing $\frac{1}{m}\sum_{i=1}^m\xi_i$ is choosing the minimal value of ξ_i for each i

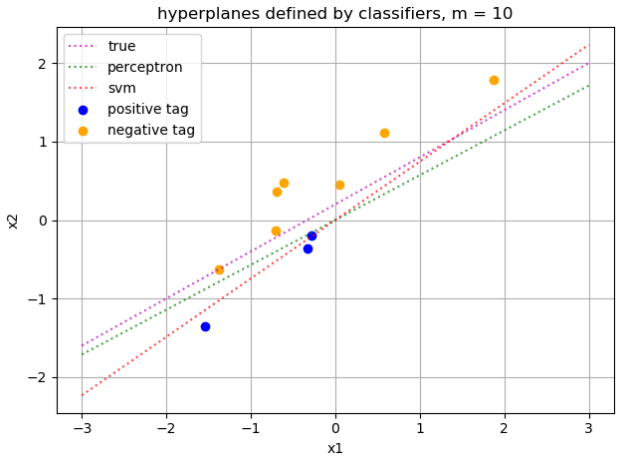
Therefore, our minimal choice of ξ_i would then be $\max(0, 1 - y_i\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle) = l^{hinge}(y_i\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle)$, for all i.

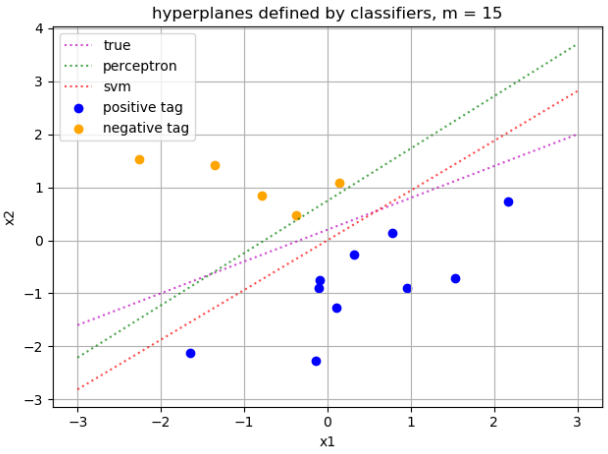
Hence we may conclude that the above expressions are equivalent, as requested ■

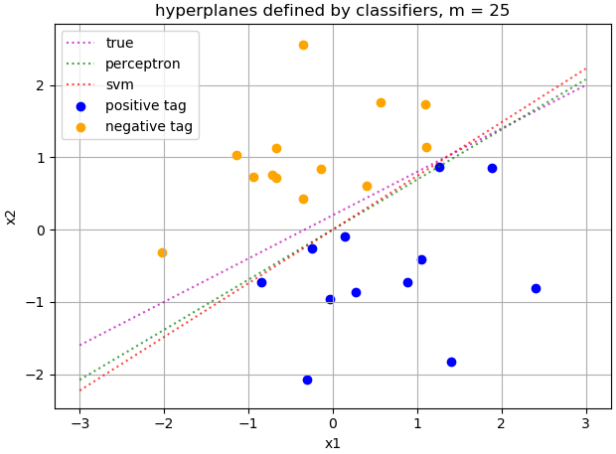
9. For each $m \in \{5, 10, 15, 25, 70\}$, draw m training points $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and create the following figure:

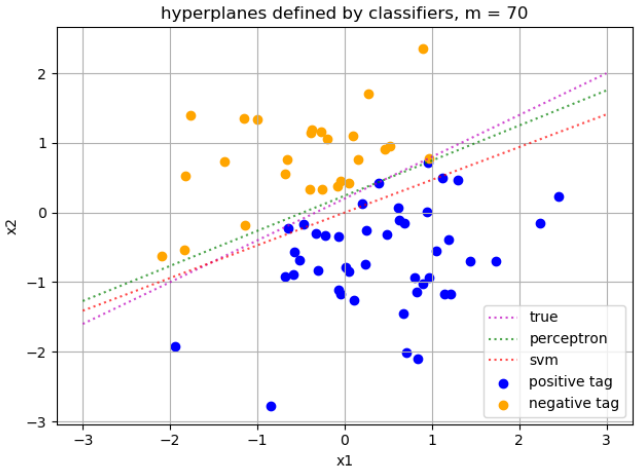
   - The drawn data points, colored according to their labels (e.g., blue for positive labels and orange for negative ones)
   - Add the hyperplane of the **true hypothesis** (the function f)
   - Add the hyperplane of the hypothesis generated by the **perceptron**
   - Add the hyperplane of the hypothesis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







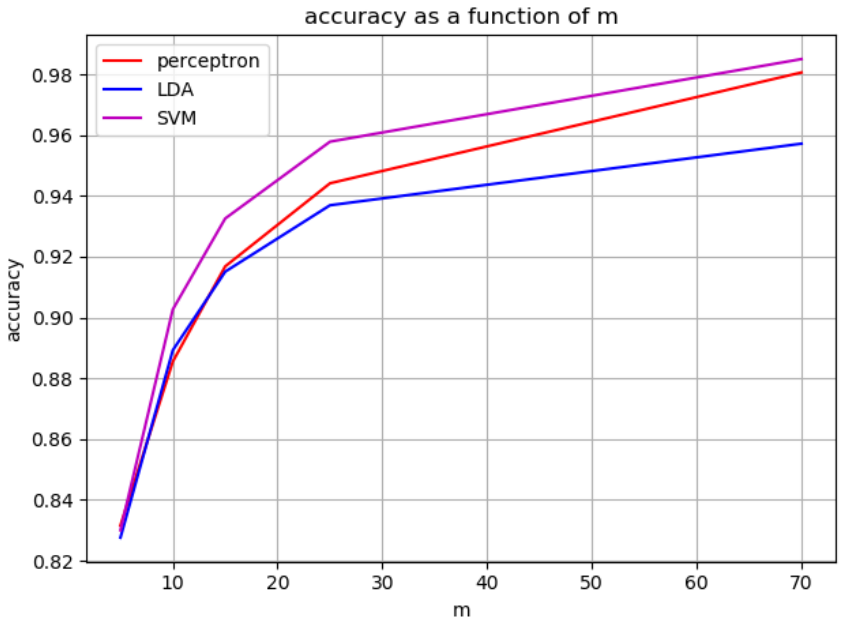




10. We'll now add a test set to compare three of the above algorithms. We'll use the following procedure:

    - 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.

It is noticeable that LDA was the least successful one, as it estimates the statistics of the sample space.

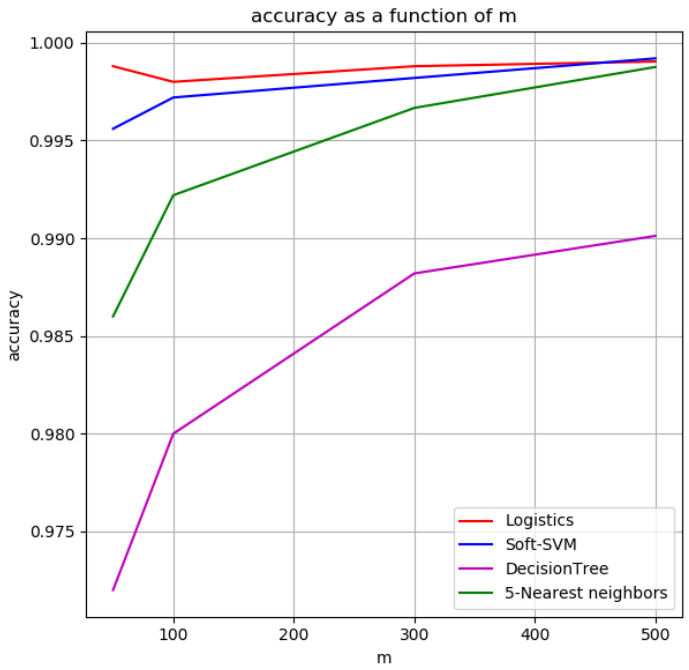
Where estimating the statistics over 70 samples only estimates the statistics to some accuracy, and the other algorithms does not relay on the statistics of the samples and converges iteratively.

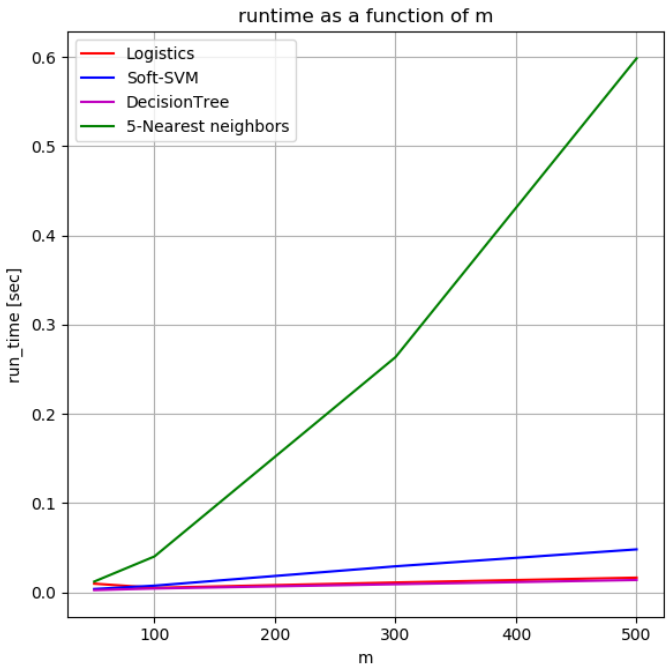
In this case, as the true class of the samples is defined by a linear equation, we may notice that the iterative algorithms converged with good accuracy to the ground truth estimation.

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.





It seems that training time grows exponentially in respect to training samples for K-Nearest Neighbors algorithm

Whereas for the rest it appears to grow linearly,

all classifiers showed good accuracy (99%+) for m=500.