CS 6780: Advanced Machine Learning **(Submitted on: 3/6/2019)**

Submission Assignment #2

Instructor: Thorsten Joachims *Name:* Molly Ingram, Julien Neves, *Netid:* msi34, jmn252

Problem 1

(a)

Figure 1 shows one possible labeling and solution where we have one $-$ and three $+$. To get every possible iteration of one $-$ and three $+$, we simply need to rotate the points and the ellipse by 90°, 180° and 270°.

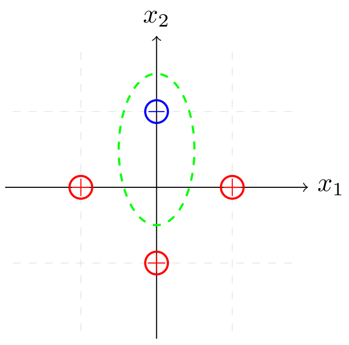


Figure 1: One $-$ and three $+$

Figure 2 shows two possible labeling and solutions where we have two $-$ and two $+$. To get every possible iteration, we simply need to rotate the first figure by 180° and flip the second figure along the both the x_1 and x_2 axes.

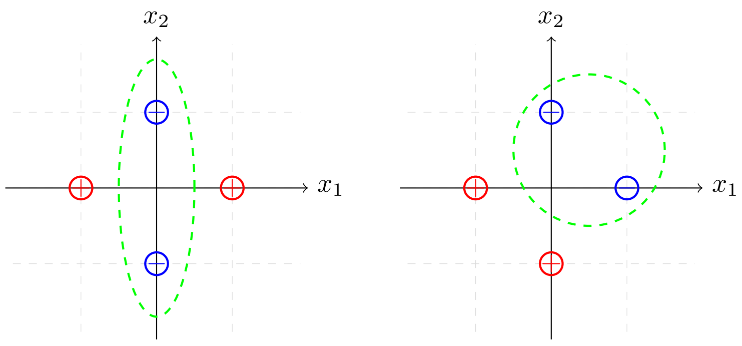


Figure 2: Two $-$ and two $+$

Figure 3 shows one possible labeling and solution where we have three $-$ and one $+$. To get every possible iteration of three $-$ and one $+$, we simply need to rotate the points and the ellipse by 90°, 180° and 270°.

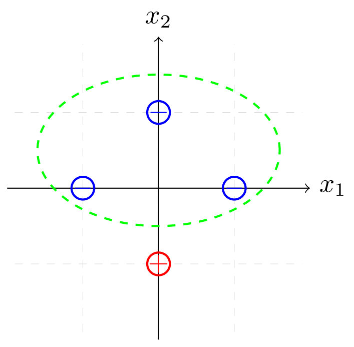


Figure 3: Three − and one +

Figure 4 shows the case where we have four − or four +.

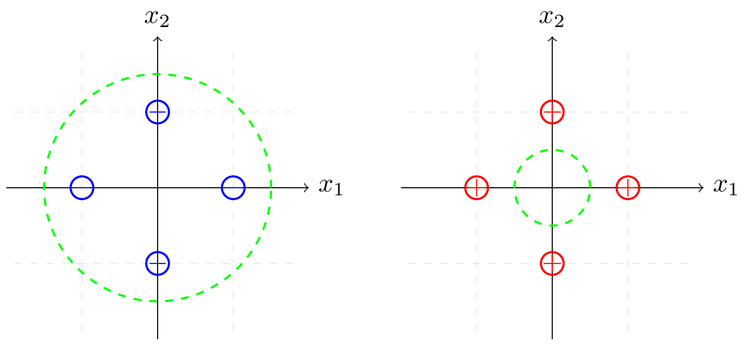


Figure 4: Four + and no −

Therefore, we can conclude that $VCdim(\mathcal{H}) \geq 4$.

(b)

Let's take the following mapping $\phi : \mathbb{R}^2 \to \mathbb{R}^4$ where

$$\phi(x_1, x_2) = (x_1, x_1^2, x_2, x_2^2)$$

Since $\phi(x) \subset \mathbb{R}^4$, we have that the $VCdim(\hat{\mathcal{H}})$ of linear hyperplanes for $\phi(x)$ is equal to 5.

The nice thing about this mapping is that a classication rule using linear hyperplanes in $\phi(x)$ is equivalent to our hypothesis class $\mathcal{H}$ in x. In fact, we can see this result by noting that the two following classifcation rules are the same:

$$y = +1 \Leftrightarrow w_1(\phi_1) + w_2(\phi_2) + w_3(\phi_3) + w_4(\phi_4) \geq b$$
$$y = +1 \Leftrightarrow w_1(x_1) + w_2(x_1)^2 + w_3(x_2) + w_4(x_2)^2 \geq b$$

Problem 2

(a)

First, note that by Cauchy-Schwarz, we have the following

$$\begin{aligned}-\|w_{\text{opt}}\|\,\|x_i\| \leq w_{\text{opt}}\cdot x_i &\leq \|w_{\text{opt}}\|\,\|x_i\| \\ -\mathcal{R} \leq w_{\text{opt}}\cdot x_i &\leq \mathcal{R} \\ -\mathcal{R}+b_{\text{opt}} \leq w_{\text{opt}}\cdot x_i + b_{\text{opt}} &\leq \mathcal{R}+b_{\text{opt}}\end{aligned}$$

Let's assume that $|b_{\text{opt}}| \geq \mathcal{R}$. Then, we have two cases: $b_{\text{opt}} \geq \mathcal{R}$ and $b_{\text{opt}} \leq -\mathcal{R}$.

For $b_{\text{opt}} \geq \mathcal{R}$, we then have

$$0 \leq w_{\text{opt}}\cdot x_i + b_{\text{opt}}$$

If $w_{\text{opt}}\cdot x_i + b_{\text{opt}} = 0$, then $\delta = 0$ which we assumed was not the case. Therefore, we have $w_{\text{opt}}\cdot x_i + b_{\text{opt}} > 0$, which implies $y_{\text{pred}} = 1$ for all i which will obviously misclassified some i if $\mathcal{S}$ is not trivial.

For $b_{\text{opt}} \geq -\mathcal{R}$, we then have

$$0 \geq w_{\text{opt}}\cdot x_i + b_{\text{opt}}$$

If $w_{\text{opt}}\cdot x_i + b_{\text{opt}} = 0$, then $\delta = 0$ which we assumed was not the case. Therefore, we have $w_{\text{opt}}\cdot x_i + b_{\text{opt}} < 0$, which implies $y_{\text{pred}} = -1$ for all i which will obviously misclassified some i if $\mathcal{S}$ is not trivial.

(b)

Let $\hat{x} = (x, 1)$ and $\hat{w} = (w, b)$. It is straightforward to see that if w_{opt} and b_{opt} optimally separates $\mathcal{S}$, then $\hat{w}_{\text{opt}} = (w_{\text{opt}}, b_{\text{opt}})$ optimally separates $\hat{\mathcal{S}}$ where $\hat{\mathcal{S}}$ is $\mathcal{S}$ with the added 1 feature since $w_{\text{opt}}\cdot x_i + b_{\text{opt}} = (w_{\text{opt}}, b_{\text{opt}})\cdot\hat{x}$.

Now, we now that for unbiased perceptron the error bound is $\frac{\hat{\mathcal{R}}^2}{\hat{\delta}^2}$ and therfore the bound for the biased perceptron by equivalence. The only thing missing is to derive $\hat{\mathcal{R}}$ and $\hat{\delta}$.

For $\hat{\mathcal{R}}$, we have the following

$$\begin{aligned}\hat{\mathcal{R}} &= \max_i \|\hat{x}_i\| \\ &= \max_i \|(x_i, 1)\| \\ &= \max_i \sqrt{x_i\cdot x_i + 1} \\ &= \max_i \sqrt{\|x_i\|^2 + 1} \\ &= \sqrt{\mathcal{R}^2 + 1}\end{aligned}$$

For δ, we know that $\delta = \min_i y_i(w_{\text{opt}}\cdot x_i + b_{\text{opt}}) = \min_i y_i(\hat{w}_{\text{opt}}\cdot\hat{x}_i)$ which would imply that $\delta = \hat{\delta}$ if $\|\hat{w}_{\text{opt}}\| = 1$ which is not the case.

Thus, to get $\hat{\delta}$, we simply need to scale $\min_i y_i(\hat{w}_{\text{opt}}\cdot\hat{x}_i)$ by $\|\hat{w}_{\text{opt}}\|$, i.e.

$$\begin{aligned}\hat{\delta} &= \frac{1}{\|\hat{w}_{\text{opt}}\|}\min_i y_i(\hat{w}_{\text{opt}}\cdot\hat{x}_i) \\ &= \frac{\delta}{\|\hat{w}_{\text{opt}}\|}\end{aligned}$$

Since by assumption $|b_{\text{opt}}| = 1$, we have that $\|\hat{w}_{\text{opt}}\| = \sqrt{w_{\text{opt}}\cdot w_{\text{opt}} + b_{\text{opt}}^2} = \sqrt{\|w_{\text{opt}}\| + 1} = \sqrt{2}$.

Therefore, the mistake bound for the biased perceptron is given by the following formula

$$\begin{aligned}\frac{\hat{\mathcal{R}}^2}{\hat{\delta}^2} &= \frac{\sqrt{\mathcal{R}^2+1}^2}{\left(\frac{\delta}{\|\hat{w}_{\text{opt}}\|}\right)^2} \\ &= \frac{2(\mathcal{R}^2+1)}{\delta^2}\end{aligned}$$

(c)

For this problem, we let $\mathcal{S}$ be the set of the following points $\{(1,1),+1\},\{(-1,1),+1\},\{(-1,-1),-1\},\{(1,-1),-1\}$. It is easy to see that $\mathcal{R}=\sqrt{2}$, while the optimal separating hyperplane is the x_1 axis, which implies that $\delta=1$. Thus, our mistake bound is 2.

Moreover, Figure 5 shows that if we are starting with the point $(1,1)$, the algorithm will make two mistakes before reaching stopping, i.e. the mistake bound is tight. Notice that for any starting point, we get the same result.

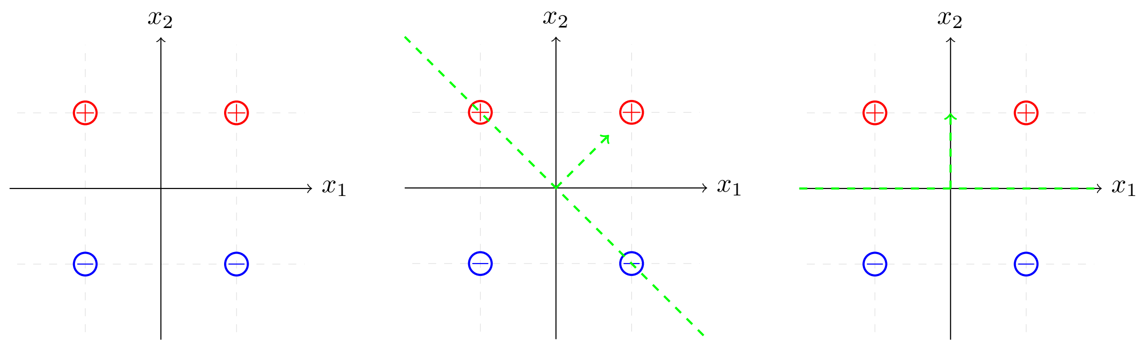


Figure 5: Example with $\frac{R^2}{\delta^2}=2$

(b)

Problem 3

(a) The batch algorithm we are minimizing is

$$F(\mathbf{w_i}) = \sum_{j=1}^{n} \max(0, -y_j(\mathbf{w_i} \cdot \mathbf{x_j}))$$

The global minimum, 0, is achieved when all n data points are classified correctly, i.e. $y_j(\mathbf{w_i} \cdot x_j) \geq 0$.

So the gradient is $\nabla_{\mathbf{w}} F(\mathbf{w_i}) = \sum_{j \in E} y_j \mathbf{x_j}$ where E is the set of all misclassified data points ($y_j(\mathbf{w_i} \cdot x_j) \leq 0$). Note the that loop in line 5 uses line 7 to sum over the misclassified points, so this algorithm is computing the gradient at $\mathbf{w_i}$.

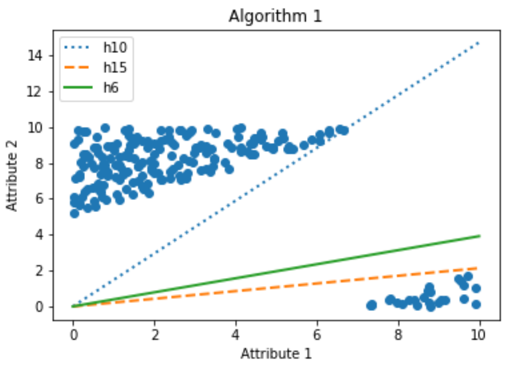
(b) Note the max element is only non-zero if $1 - y_j(\mathbf{w} \cdot \mathbf{x_j}) \geq 0 \leftrightarrow 1 \geq y_j(\mathbf{w} \cdot \mathbf{x_j})$. Let E be that set of all data points such that this non-zero condition is satisfied. Note this set includes misclassified points *and* correctly classified points that are close to the hyperplane. Taking the partial derivate of F wrt the i^{th} element of $\mathbf{w}$ we have

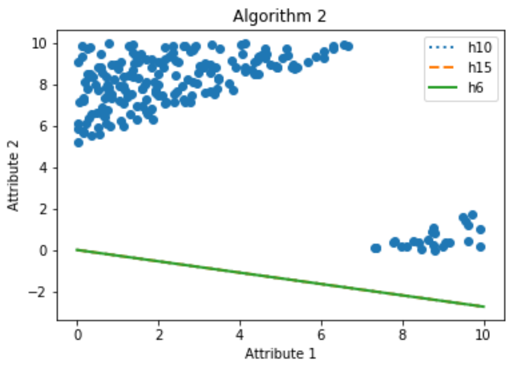
$$F_i(\mathbf{w}) = w_i - C \sum_{j \in E} y_j x_{ji}$$

Stacking the partials, we can rewrite in the vector form $\nabla_{\mathbf{w}} F(\mathbf{w}) = \mathbf{w} - C \sum_{j \in E} y_j \mathbf{x_j}$.

The algorithm is updated as i) $\mathbf{w_i}$ where i indicates the iteration; ii) $1 \geq y_j(\mathbf{w_i} \cdot \mathbf{x_j}$; iii) $\Delta \mathbf{w_{temp}} - C y_j \mathbf{x_j}$

(c) The difference in the update condition is that algorithm 1 only uses points incorrectly classified and algorithm 2 uses misclassified points *and* correctly classified points that are close to the hyperplane. So algorithm 2 includes more in examples in an update. The order of the dataset will not affect the algorithms because they loop over all points when they update, not update one point at a time like the perceptron algorithm we discussed in lecture.





(d)

I have plotted only some of the hyperplanes with different α values for clarity. We can see from the scatter plots that algorithm 1 is more sensitive to α and that it allows the hyperplane to come closer to the examples. Algorithm 2 is less sensitive to α, the hyperplane essentially overlap, and perhaps because of the margins and the few number of points in the bottom right the hyperplane lumps all points in the same classification. ...It also could be I made an error in the algorithm because this doesn't seem right...

The table of weight vectors, shown in the jupyter notebook along with the code, also indicates the algorithm 2 is much less sensitive to α.

(e) On the unsorted examples, the number of updates was 93 and the number of passes was 78. On the sorted examples, The number of updates was 441 and the number of passes was 397. Because the algorithm runs over the examples in order until a separating hyperplane is found, the sorted data groups together those which should not be separated so we have to cover more examples before finding an example that is classifed differently. While in the unsorted data, the algorithm gets to examples classified differently much earlier in looping over the examples.