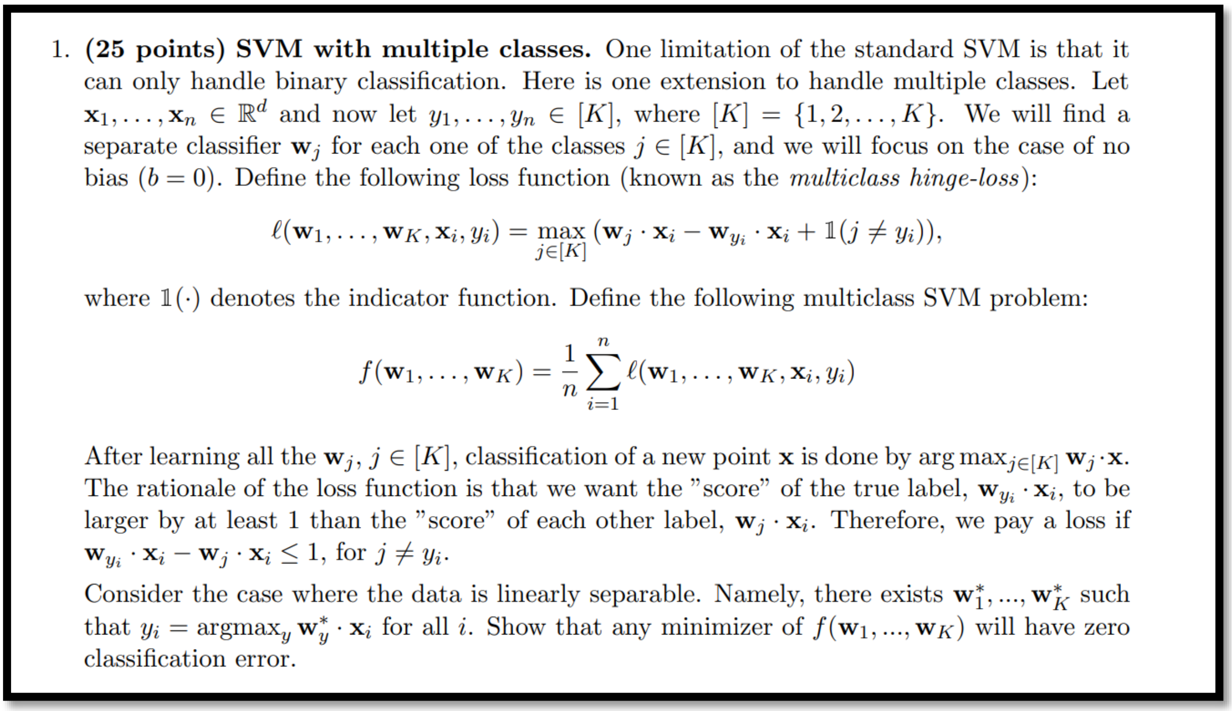
Introduction to Machine Learning-Exercise 4

Odai Agbaria, Id: 212609440

Theory Questions



1. **(25 points) SVM with multiple classes.** One limitation of the standard SVM is that it can only handle binary classification. Here is one extension to handle multiple classes. Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and now let $y_1, \ldots, y_n \in [K]$, where $[K] = \{1, 2, \ldots, K\}$. We will find a separate classifier $\mathbf{w}_j$ for each one of the classes $j \in [K]$, and we will focus on the case of no bias ($b = 0$). Define the following loss function (known as the *multiclass hinge-loss*):

$$\ell(\mathbf{w}_1, \ldots, \mathbf{w}_K, \mathbf{x}_i, y_i) = \max_{j \in [K]} (\mathbf{w}_j \cdot \mathbf{x}_i - \mathbf{w}_{y_i} \cdot \mathbf{x}_i + \mathbb{1}(j \neq y_i)),$$

where $\mathbb{1}(\cdot)$ denotes the indicator function. Define the following multiclass SVM problem:

$$f(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{w}_1, \ldots, \mathbf{w}_K, \mathbf{x}_i, y_i)$$

After learning all the $\mathbf{w}_j$, $j \in [K]$, classification of a new point $\mathbf{x}$ is done by $\arg\max_{j \in [K]} \mathbf{w}_j \cdot \mathbf{x}$. The rationale of the loss function is that we want the "score" of the true label, $\mathbf{w}_{y_i} \cdot \mathbf{x}_i$, to be larger by at least 1 than the "score" of each other label, $\mathbf{w}_j \cdot \mathbf{x}_i$. Therefore, we pay a loss if $\mathbf{w}_{y_i} \cdot \mathbf{x}_i - \mathbf{w}_j \cdot \mathbf{x}_i \leq 1$, for $j \neq y_i$.

Consider the case where the data is linearly separable. Namely, there exists $\mathbf{w}_1^*, \ldots, \mathbf{w}_K^*$ such that $y_i = \text{argmax}_y \, \mathbf{w}_y^* \cdot \mathbf{x}_i$ for all i. Show that any minimizer of $f(\mathbf{w}_1, \ldots, \mathbf{w}_K)$ will have zero classification error.

Solution:

First, we will show that the minimizer of $f(w_1, \ldots, w_k)$ holds that f=0, and then we will show that any $w \in R^k, which\ holds\ that\ f(w) = 0\ then\ it\ has\ zero\ classification\ error.$

Notice that by definition of f it holds that:

$$\begin{aligned} min_{\,w_1,\ldots,w_k} f(w_1, \ldots, w_k) &= min_{\,w_1,\ldots,w_k} \frac{1}{n} \sum_{i=1}^{n} l(w_1, \ldots, w_k, x_i, y_i) \\ &= min_{\,w_1,\ldots,w_k} \frac{1}{n} \sum_{i=1}^{n} \max_{j \in [K]} \left(w_j * x_i - w_{y_i} * x_i + I(j \neq y_i) \right) \\ &= min_{\,w_1,\ldots,w_k} \frac{1}{n} \sum_{i=1}^{n} \max\left(\max_{j \in [K], \neq y_i} (w_j * x_i - w_{y_i} * x_i + 1), 0 \right) \end{aligned}$$

Meaning that by definition of f it holds that f is non-negative.

So, if we show that there exists $(w_1, \ldots, w_k)$ such that $f(w_1, \ldots, w_k) = 0$ then the minimizer must hold that too.

Notice that the data is linearly separable, meaning there exists $(w_1', \ldots, w_k')$ such that $y_i = argmax_y w_y' * x_i$ for every i. Denote $w' = (w_1', \ldots, w_k')$.

Consider that w'. If for every i it holds that:

$$for\ every\ j \neq i: w'_{y_i} * x_i - w'_j * x_i \geq 1 \quad ★$$

Then $\sum_{i=1}^{n} \max\left(\max_{j\in[K],\neq y_i}(w'_j * x_i - w'_{y_i} * x_i + 1),0\right) =$ $\sum_{i=1}^{n} \max\left(\max_{j\in[K],\neq y_i}(negative),0\right) = \sum_{i=1}^{n} \max(\text{negative}, 0) = \sum_{i=1}^{n} 0 = 0$. Meaning that in this case $f(w') = 0$ and thus it minimizes f since $f \geq 0$.

If ★ doesn't hold for every i: then there exists at least one i such that:

$$there\ exists\ j \neq i: w'_{y_i} * x_i - w'_j * x_i < 1$$

Notice that $w'_{y_i} * x_i - w'_j * x_i \geq 0$ since we are in the realizable case(since $y_i = argmax_y w_y{}' * x_i$).

Denote $m = min_{i,j\neq i}(w'_{y_i} * x_i - w'_j * x_i)$, $0 < m < 1$ as we can conclude from the two lines above.

Then consider $w'' = \frac{1}{m}(w_1{}', \dots, w_k{}')$. Thus, for every i it holds that:

$$for\ every\ j \neq i: w''_{y_i} * x_i - w''_j * x_i = \frac{w'_{y_i}}{m} * x_i - \frac{w'_j}{m} * x_i = \frac{1}{m}\left(w'_{y_i} * x_i - w'_j * x_i\right) \geq \frac{1}{m} * m = 1$$

Thus also here $\sum_{i=1}^{n} \max\left(\max_{j\in[K],\neq y_i}(w''_j * x_i - w''_{y_i} * x_i + 1),0\right) =$ $\sum_{i=1}^{n} \max\left(\max_{j\in[K],\neq y_i}(negative),0\right) = \sum_{i=1}^{n} \max(\text{negative}, 0) = \sum_{i=1}^{n} 0 = 0$. Meaning that in this case $f(w'') = 0$ and thus it minimizes f since $f \geq 0$.

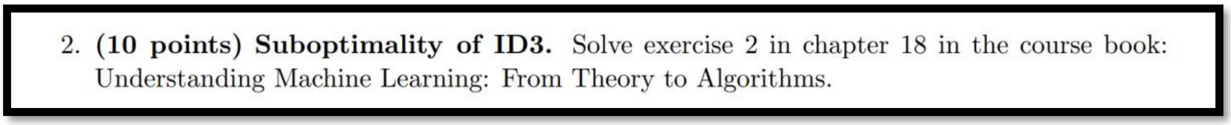
Till this point we have proved that the minimizer of f holds that f=0.

Now notice that $f(w) = f(w_1, \dots, w_k) = 0$ $\boldsymbol{iff}$ the score of the true label, $w_{y_i} * x_i$, is larger by at least 1 than the score of each other label, $w_j * x_i$, and then the classification of a point x_i is done by:

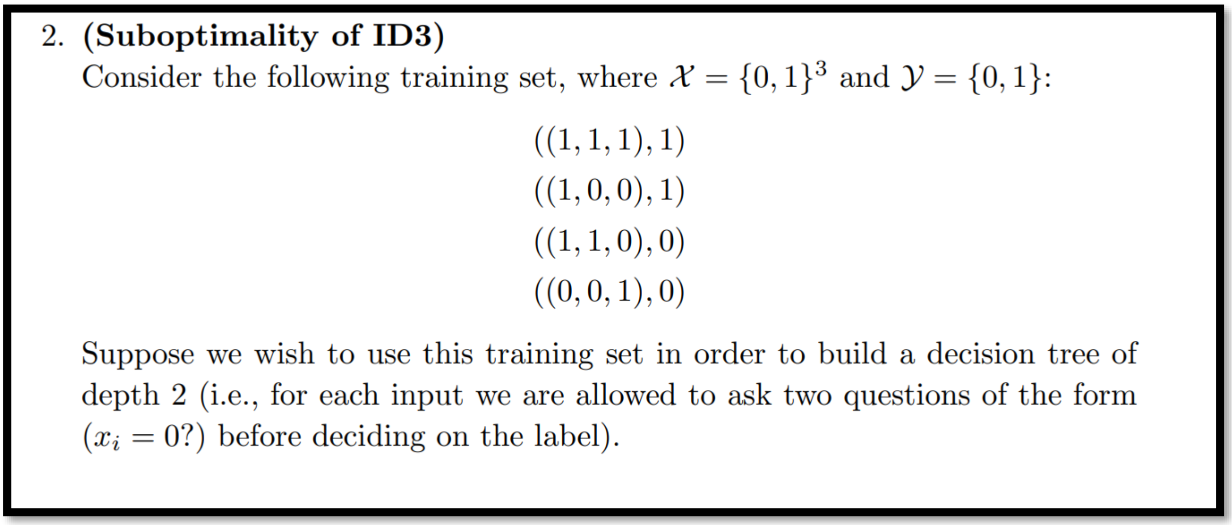
$argmax_{j\in[K]} w_j * x_i$ which is equal to y_i since as i said above $w_{y_i} * x_i$ is the greater. And thus w has no error, it classifies the data points perfectly.

To conclude, we have proved that w the minimizer of f holds that $f(w) = 0$ and then we showed that such a w has zero classification error.

■



2. **(10 points) Suboptimality of ID3.** Solve exercise 2 in chapter 18 in the course book: Understanding Machine Learning: From Theory to Algorithms.



2. **(Suboptimality of ID3)**
Consider the following training set, where $\mathcal{X} = \{0,1\}^3$ and $\mathcal{Y} = \{0,1\}$:

$$((1,1,1),1)$$
$$((1,0,0),1)$$
$$((1,1,0),0)$$
$$((0,0,1),0)$$

Suppose we wish to use this training set in order to build a decision tree of depth 2 (i.e., for each input we are allowed to ask two questions of the form $(x_i = 0?)$ before deciding on the label).

1. Suppose we run the ID3 algorithm up to depth 2 (namely, we pick the root node and its children according to the algorithm, but instead of keeping on with the recursion, we stop and pick leaves according to the majority label in each subtree). Assume that the subroutine used to measure the quality of each feature is based on the entropy function (so we measure the *information gain*), and that if two features get the same score, one of them is picked arbitrarily. Show that the training error of the resulting decision tree is at least 1/4.

Solution:

At first $S = \{((1,1,1),1),((1,0,0),1),((1,1,0),0),((0,0,1),0)\}$.

Let's run the ID3 algorithm up to depth 2 as we saw in recitations.

First, we calculate G(S,1), G(S,2) and G(S,3) to choose the first predicate at the root:

$$G(S,i) = entropy\big(P_S(Y=1)\big) - \big(P_S(X_i=1) * entropy\big(P_S(Y=1|X_i=1)\big) + P_S(X_i=0) * entropy\big(P_S(Y=1|X_i=0)\big)\big)$$

$$G(S,1) = entropy\left(\frac{1}{2}\right) - \left(\frac{3}{4} * entropy\left(\frac{2}{3}\right) + \frac{1}{4} * entropy(0)\right) = \frac{1}{2} - \left(\frac{3}{4} * 0.459 + \frac{1}{4} * 0\right) = 0.15575$$

$$G(S,2) = entropy\left(\frac{1}{2}\right) - \left(\frac{1}{2} * entropy\left(\frac{1}{2}\right) + \frac{1}{2} * entropy\left(\frac{1}{2}\right)\right) = \frac{1}{2} - \left(\frac{1}{2} * \frac{1}{2} + \frac{1}{2} * \frac{1}{2}\right) = 0$$

$$G(S,3) = entropy\left(\frac{1}{2}\right) - \left(\frac{1}{2} * entropy\left(\frac{1}{2}\right) + \frac{1}{2} * entropy\left(\frac{1}{2}\right)\right) = \frac{1}{2} - \left(\frac{1}{2} * \frac{1}{2} + \frac{1}{2} * \frac{1}{2}\right) = 0$$

Since $G(S,1)$ is the largest among the three predicates then we choose this predicate, meaning we first at the root choose the predicate: $x_1 = 0?$

Then notice that if $x_1 = 0$ then $S = \{((0,0,1),0)\}$ so in this case we got a leaf with the label 0.

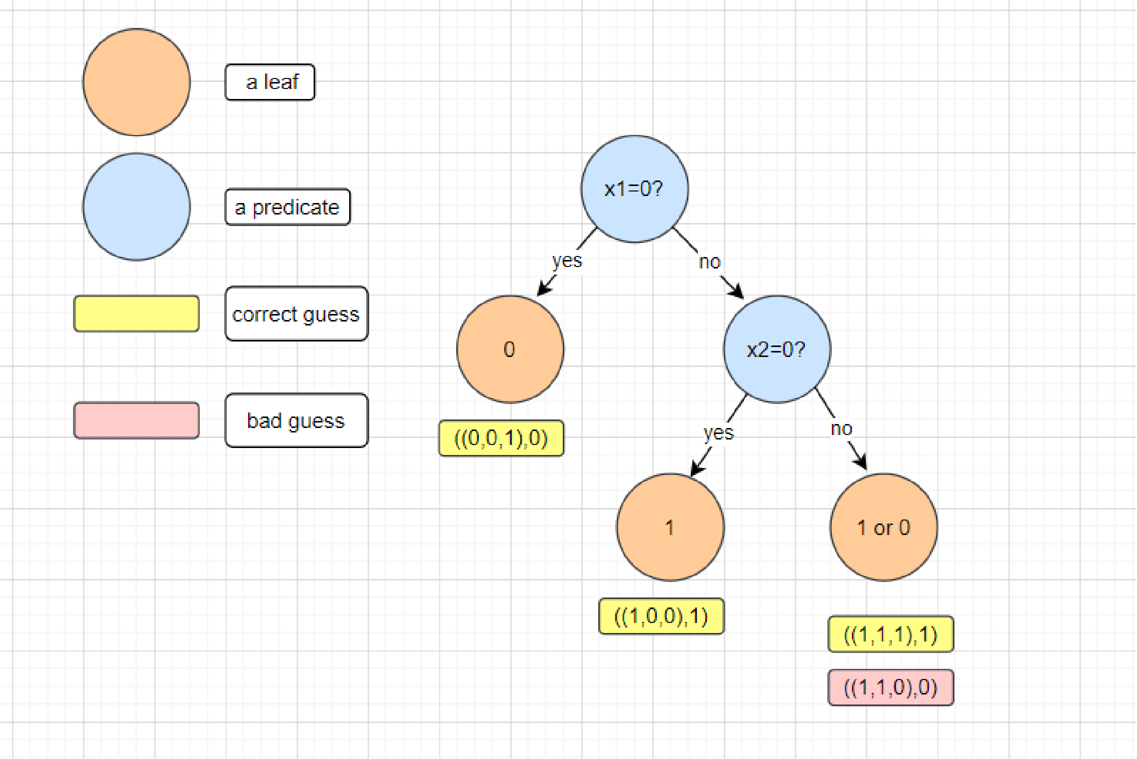
In the other case, $x_1 = 1$, then we get: $S = \{((1,1,1),1),((1,0,0),1),((1,1,0),0)\}$ so now we need to choose one of the predicates: $x_2 = 0?\, x_3 = 0?$ Let's calculate the gain:

$$G(S,2) = entropy\left(\frac{2}{3}\right) - \left(\frac{2}{3} * entropy\left(\frac{1}{2}\right) + \frac{1}{3} * entropy(1)\right) = entropy\left(\frac{2}{3}\right) - \left(\frac{2}{3} * \frac{1}{2} + \frac{1}{3} * 0\right) = entropy\left(\frac{2}{3}\right) - \frac{1}{3}$$

$$G(S,3) = entropy\left(\frac{2}{3}\right) - \left(\frac{1}{3} * entropy(1) + \frac{2}{3} * entropy\left(\frac{1}{2}\right)\right) = entropy\left(\frac{2}{3}\right) - \left(\frac{1}{3} * 0 + \frac{2}{3} * \frac{1}{2}\right) = entropy\left(\frac{2}{3}\right) - \frac{1}{3}$$

So they has the same gain so we are going to choose one predicate of them, and then we will get a tree of depth 2 so we will stop as w eare asked in the question and assign the label in the leaves due to the majority:

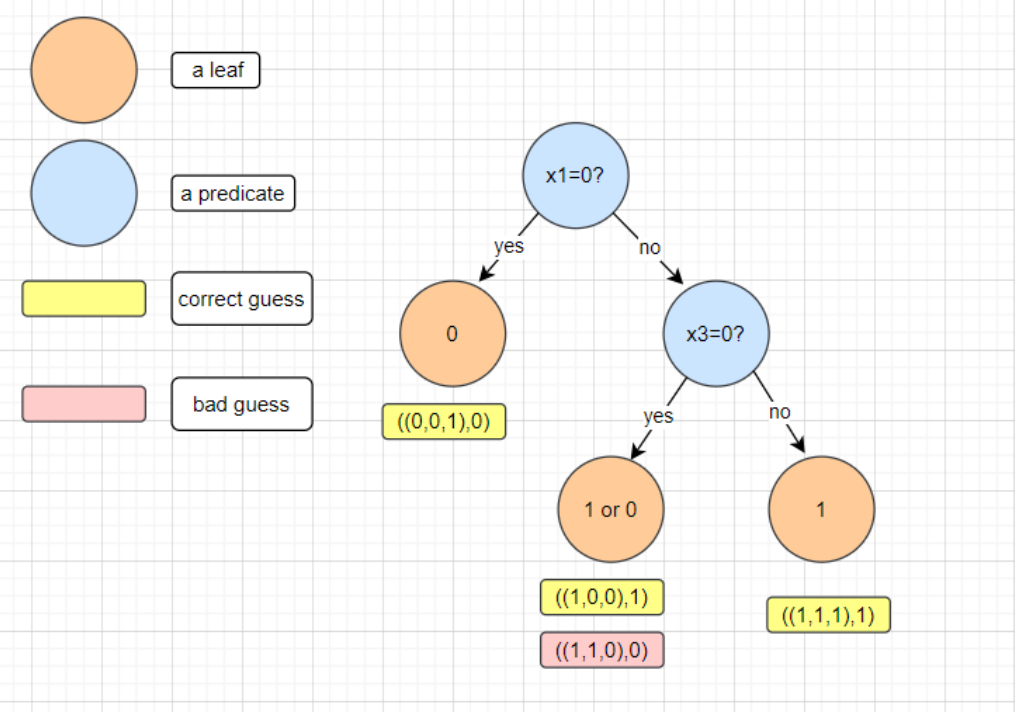
If we choose $x_2 = 0?$ as the second predicate, we get the next tree:



Notice that in case x2=1 in the second predicate then we remain with two data points one with label 1 and one with label 0, so this leaf could return 1 or 0(they both are the majority label), but that doesn't matter as in every situation we will get one data point wrongly (fail in it's prediction).

So, in this case our resulting tree fails at one data point out of 4, then its training error is $\frac{1}{4}$.

If we choose $x_3 = 0?$ as the second predicate, we get the next tree:



Notice that also here in case x3=0 in the second predicate then we remain with two data points one with label 1 and one with label 0, so this leaf could return 1 or 0(they both are the majority label), but that doesn't matter as in every situation we will get one data point wrongly (fail in it's prediction).

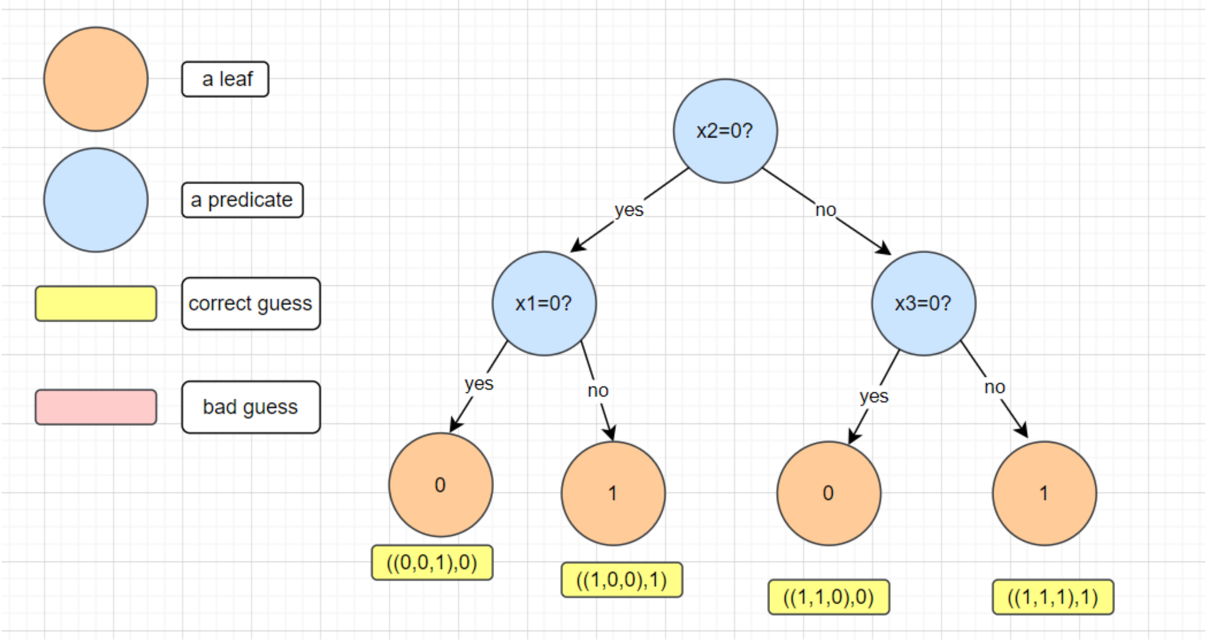
So, also in this case our resulting tree fails at one data point out of 4, then its training error is $\frac{1}{4}$.

Hence, we get that the training error of the resulting decision tree is at least $\frac{1}{4}$.

2. Find a decision tree of depth 2 that attains zero training error.

Solution:

Consider the next decision tree of depth 2 which attains zero training error:



Notice that the ID3 algorithm won't get to this tree since the algorithm always chooses the predicate x1=0? At the root because it's gain is larger than the other.

3. **(25 points) Step-size Perceptron.** Consider the modification of Perceptron algorithm with the following update rule:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t y_t \mathbf{x}_t$$

whenever $\hat{y}_t \neq y_t$ ($\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$ otherwise). Assume that data is separable with margin $\gamma > 0$ and that $\|\mathbf{x}_t\| = 1$ for all t. For simplicity assume that the algorithm makes M mistakes at the first M rounds, after which it makes no mistakes. For $\eta_t = \frac{1}{\sqrt{t}}$, show that the number of mistakes step-size Perceptron makes is at most $\frac{4}{\gamma^2}\log(\frac{1}{\gamma})$. (Hint: use the fact that if $x \leq a\log(x)$ then $x \leq 2a\log(a)$). It's okay if you obtain a bound with slightly different constants, but the asymptotic dependence on γ should be tight.

Solution:

The proof will be similar to the one we did in class but with some changes.

Denote S the sequence of the labelled examples consistent with linear prediction $x \rightarrow sign(w^{*}x)$ for some w^{*} of unit length, $R = \max_{x \in S} ||x||$.

Lemma 1: If mistake was made at iteration t then: $w_{t+1} * w^{*} \geq w_t * w^{*} + \eta_t \gamma$.

Proof: assume that a mistake is made at iteration t then:

$$w_{t+1} * w^{*} = (w_t + \eta_t y_t x_t) * w^{*} = w_t w^{*} + \eta_t y_t x_t w^{*}$$

Notice that by the definition of $\gamma \coloneqq \min_{(x,y) \epsilon S} y w^{*} x$ then it holds that $y_t x_t w^{*} \geq \gamma$ thus we get:

$$w_{t+1} * w^{*} = w_t w^{*} + \eta_t y_t x_t w^{*} \geq w_t w^{*} + \eta_t \gamma$$

(I don't see any necessity to split according to y_t,if it's 1 or -1, as we did in class since $y_t x_t w^{*} \geq \gamma$ holds for both cases)

■

Lemma 2: If mistake was made at iteration t then: $||w_{t+1}||^2 \leq \left||w_t|\right|^2 + {\eta_t}^2 R^2$.

Proof: assume that a mistake is made at iteration t then:

$$||w_{t+1}||^2 = ||w_t + \eta_t y_t x_t||^2 = ||w_t||^2 + ||\eta_t y_t x_t||^2 + 2\eta_t y_t x_t w_t$$
$$= ||w_t||^2 + {\eta_t}^2 {y_t}^2 ||x_t||^2 + 2\eta_t y_t x_t w_t$$

If $y_t = 1$ we get:

$$||w_{t+1}||^2 = ||w_t||^2 + {\eta_t}^2 1^2 ||x_t||^2 + 2\eta_t * 1 * x_t w_t = ||w_t||^2 + {\eta_t}^2 ||x_t||^2 + 2\eta_t x_t w_t$$

Notice that by definition of R$= \max_{x \epsilon S} ||x||$, then $||x_t||^2 \leq R^2$, and that since we made a mistake in iteration t then $sign(x_t w_t) \neq y_t = 1 \rightarrow sign(x_t w_t) < 0$ and thus we get:

$$||w_{t+1}||^2 = ||w_t||^2 + {\eta_t}^2 ||x_t||^2 + \underbrace{2\eta_t x_t w_t}_{\text{Negative expression}} \leq \left||w_t|\right|^2 + {\eta_t}^2 \left||x_t|\right|^2 \leq \left||w_t|\right|^2 + {\eta_t}^2 R^2$$

If $y_t = -1$ we get:

$$||w_{t+1}||^2 = ||w_t||^2 + {\eta_t}^2 (-1)^2 ||x_t||^2 + 2\eta_t * -1 * x_t w_t = \left||w_t|\right|^2 + {\eta_t}^2 \left||x_t|\right|^2 - 2\eta_t x_t w_t$$

Notice that also here by definition of R$= \max_{x \epsilon S} ||x||$, then $||x_t||^2 \leq R^2$ and that since we made a mistake in iteration t then $sign(x_t w_t) \neq y_t = -1 \rightarrow sign(x_t w_t) > 0$ and thus we get:

$$||w_{t+1}||^2 = \left||w_t|\right|^2 + {\eta_t}^2 \left||x_t|\right|^2 \underbrace{- 2\eta_t x_t w_t}_{\text{Negative expression}} \leq \left||w_t|\right|^2 + {\eta_t}^2 \left||x_t|\right|^2 \leq \left||w_t|\right|^2 + {\eta_t}^2 R^2$$

■

By Lemma 1 we get that after M mistakes $w_t * w^{*} \geq \sum_{t=1}^{M} \eta_t \gamma = \gamma \sum_{t=1}^{M} \eta_t$ and that's due to that every time we do a mistake the response increases with at least $\eta_t \gamma$.

By Lemma 2 we get that after M mistakes $||w_t||^2 \leq \sum_{t=1}^{M} {\eta_t}^2 R^2 = R^2 \sum_{t=1}^{M} {\eta_t}^2$ and that's due to that every time we do a mistake the norm increases with at most ${\eta_t}^2 R^2$.

By Cauchy-Shwarz inequality we get that: $w_t * w^{*} \leq \left||w_t|\right| * \left||w^{*}|\right|$ and since w^{*} is of unit length then we get $w_t * w^{*} \leq \left||w_t|\right|$.

Thus, we got:

$$\gamma\sum_{t=1}^{M}\eta_t \le w_t * w^* \le ||w_t|| \le \sqrt{R^2\sum_{t=1}^{M}{\eta_t}^2} = R\sqrt{\sum_{t=1}^{M}{\eta_t}^2}$$

In particular we got:

$$\gamma\sum_{t=1}^{M}\eta_t \le R\sqrt{\sum_{t=1}^{M}{\eta_t}^2}$$

We know that $\eta_t = \frac{1}{\sqrt{t}}$:

$$\gamma\sum_{t=1}^{M}\frac{1}{\sqrt{t}} \le R\sqrt{\sum_{t=1}^{M}\frac{1}{\sqrt{t}}^2} = R\sqrt{\sum_{t=1}^{M}\frac{1}{t}} \quad \rightarrow \sum_{t=1}^{M}\frac{1}{\sqrt{t}} \le \frac{R}{\gamma}\sqrt{\sum_{t=1}^{M}\frac{1}{t}}$$

We also know that $||x_t|| = 1$ so R=1:

$$\boldsymbol{\sum_{t=1}^{M}\frac{1}{\sqrt{t}} \le \frac{1}{\gamma}\sqrt{\sum_{t=1}^{M}\frac{1}{t}}}$$

Notice that $\sum_{t=1}^{M}\frac{1}{\sqrt{t}} \ge \sum_{t=1}^{M}\frac{1}{\sqrt{M}} = M * \frac{1}{\sqrt{M}} = \sqrt{M}$

Also from calculus(the harmonic series $\sum_{t=1}^{M}\frac{1}{t}$ can be approximated by the integral of $\frac{1}{x}$ from 1 to M which is log(M), also from mathematical analysis we know that: $\sum_{t=1}^{M}\frac{1}{t}$ is less than or equal to log(M) +0.577...) we know that $\sqrt{\sum_{t=1}^{M}\frac{1}{t}} \le \sqrt{\log(M) + 1}$

For large enough M it holds that: $\sqrt{\log(M) + 1} \le \sqrt{\log(M) + \log(M)} = \sqrt{2\log(M)}$

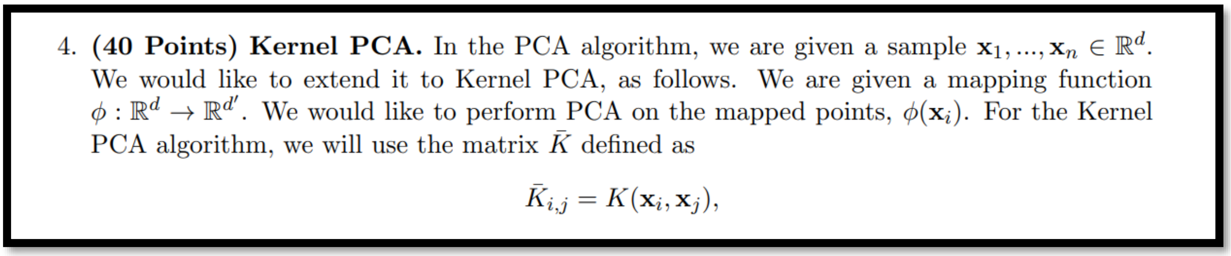
So for large enough M we get:

$$\sqrt{M} \le \sum_{t=1}^{M}\frac{1}{\sqrt{t}} \le \frac{1}{\gamma}\sqrt{\sum_{t=1}^{M}\frac{1}{t}} \le \frac{1}{\gamma}\sqrt{\log(M) + 1} \le \frac{1}{\gamma}\sqrt{2\log(M)}$$

Meaning we got that: $\sqrt{M} \le \frac{1}{\gamma}\sqrt{2\log(M)} \quad and\ thus\ \ M \le \frac{2}{\gamma^2}\log(M)$. Now, using the hint provided in the question we get:

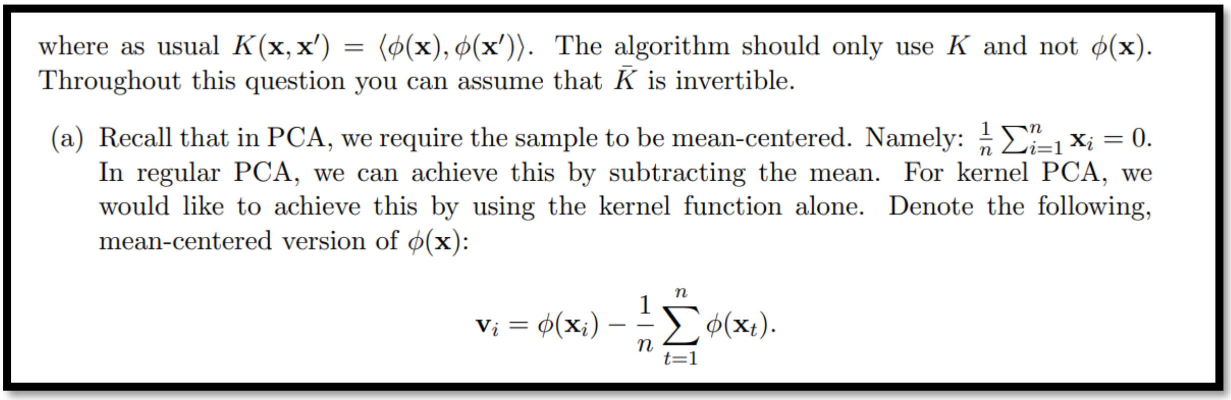
$$M \le \frac{2}{\gamma^2}\log(M) \quad \rightarrow \quad M \le \frac{4}{\gamma^2}\log\left(\frac{2}{\gamma^2}\right) \le \frac{4}{\gamma^2}\log\left(\frac{4}{\gamma^2}\right) = \frac{4}{\gamma^2}\log\left(\left(\frac{2}{\gamma}\right)^2\right) = \frac{4 * 2}{\gamma^2}\log\left(\frac{2}{\gamma}\right)$$
$$\rightarrow \quad M \le \frac{8}{\gamma^2}\log\left(\frac{2}{\gamma}\right)$$

To sum up, we showed that the number of mistakes step-size Perceptron make is at most $\frac{8}{\gamma^2}\log\left(\frac{2}{\gamma}\right)$ (the bound is with slightly different constants but the asymptotic dependence on γ is tight). ■



4. **(40 Points) Kernel PCA.** In the PCA algorithm, we are given a sample $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^d$. We would like to extend it to Kernel PCA, as follows. We are given a mapping function $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$. We would like to perform PCA on the mapped points, $\phi(\mathbf{x}_i)$. For the Kernel PCA algorithm, we will use the matrix $\bar{K}$ defined as

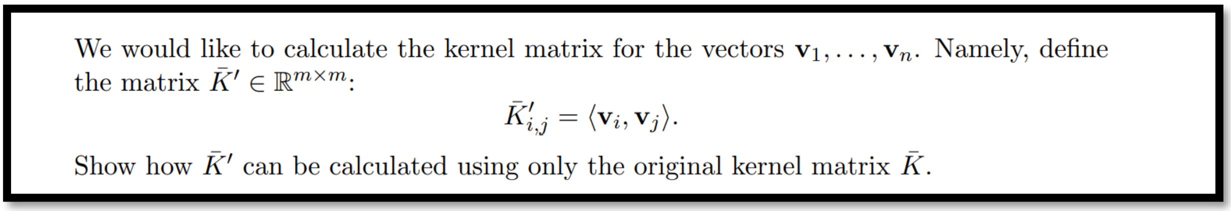
$$\bar{K}_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j),$$



where as usual $K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$. The algorithm should only use K and not $\phi(\mathbf{x})$. Throughout this question you can assume that $\bar{K}$ is invertible.

(a) Recall that in PCA, we require the sample to be mean-centered. Namely: $\frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i = 0$. In regular PCA, we can achieve this by subtracting the mean. For kernel PCA, we would like to achieve this by using the kernel function alone. Denote the following, mean-centered version of $\phi(\mathbf{x})$:

$$\mathbf{v}_i = \phi(\mathbf{x}_i) - \frac{1}{n}\sum_{t=1}^{n} \phi(\mathbf{x}_t).$$



We would like to calculate the kernel matrix for the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Namely, define the matrix $\bar{K}' \in \mathbb{R}^{m \times m}$:

$$\bar{K}'_{i,j} = \langle \mathbf{v}_i, \mathbf{v}_j \rangle.$$

Show how $\bar{K}'$ can be calculated using only the original kernel matrix $\bar{K}$.

Solution:

$$\bar{K}_{i,j} = K(x_i, x_j) = < \emptyset(x_i), \emptyset(x_j) >$$

$$\bar{K}'_{i,j} =< v_i, v_j >$$

We will show that for every i,j $\bar{K}'_{i,j}$ can be calculated using only the original kernel matrix $\bar{K}$.

It holds that:

$$\bar{K}'_{i,j} =< v_i, v_j >=< \emptyset(x_i) - \frac{1}{n}\sum_{t=1}^{n} \emptyset(x_t), \emptyset(x_j) - \frac{1}{n}\sum_{t=1}^{n} \emptyset(x_t) >$$

$$= < \emptyset(x_i), \emptyset(x_j) - < \emptyset(x_i), \frac{1}{n}\sum_{t=1}^{n} \emptyset(x_t) > - < \frac{1}{n}\sum_{t=1}^{n} \emptyset(x_t), \emptyset(x_j) > + < \frac{1}{n}\sum_{t=1}^{n} \emptyset(x_t), \frac{1}{n}\sum_{t=1}^{n} \emptyset(x_t) >$$

$$= < \emptyset(x_i), \emptyset(x_j)$$

$$> - \frac{1}{n}\sum_{t=1}^{n} < \emptyset(x_i), \emptyset(x_t) > - \frac{1}{n}\sum_{t=1}^{n} < \emptyset(x_t), \emptyset(x_j) >$$

$$+ \frac{1}{n^2}\sum_{t=1}^{n}\sum_{k=1}^{n} < \emptyset(x_t), \emptyset(x_k) >$$

$$= K(x_i, x_j) - \frac{1}{n}\sum_{t=1}^{n} K(x_i, x_t) - \frac{1}{n}\sum_{t=1}^{n} K(x_t, x_j) + \frac{1}{n^2}\sum_{t=1}^{n}\sum_{k=1}^{n} K(x_t, x_k)$$

$$= \overline{K}_{i,j} - \frac{1}{n}\sum_{t=1}^{n} \overline{K}_{i,t} - \frac{1}{n}\sum_{t=1}^{n} \overline{K}_{t,j} + \frac{1}{n^2}\sum_{t=1}^{n}\sum_{k=1}^{n} \overline{K}_{t,k}$$

So, we got that $\overline{K}'_{i,j}$ can be calculated using only the original kernel matrix $\overline{K}$. ■

(b) For the rest of the question, you may assume that $\sum_{i=1}^{n} \phi(\mathbf{x}_i) = 0$. We would like to apply PCA to the vectors $\phi(\mathbf{x}_i)$. Denote by $\mathbf{u}_1, \ldots, \mathbf{u}_k$ the first k principal components in $\mathbb{R}^{d'}$, corresponding to the sample $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)$. Assuming $k \leq n$, show that $\mathbf{u}_j$ (for $j = 1, \ldots, k$) is a linear combination of $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)$. That is, show that $\mathbf{u}_j = \sum_{i=1}^{n} \alpha_{j,i}\phi(\mathbf{x}_i)$. (Hint: use the fact that $\mathbf{w}\mathbf{w}^T\mathbf{v} = (\mathbf{w}^T\mathbf{v})\mathbf{w}$ for any vectors $\mathbf{v}$ and $\mathbf{w}$).

Solution:

Denote by $u_1, \ldots, u_k$ the first k principal components in $R^{d'}$ corresponding to the sample $\emptyset(x_1), \ldots, \emptyset(x_n)$.

We saw in class that the solution to the PCA problem are eigenvectors of Σ that correspond to the largest k eigenvalues.

In our case, applying PCA algorithm to the vectors $\emptyset(x_i)$, then we can define $\Sigma = \frac{1}{n}\sum_{i=1}^{n} \emptyset(x_i)\emptyset(x_i)^T$ and thus $u_1, \ldots, u_k$ are the eigenvectors which corresponds to the largest k eigenvalues $\lambda_1, \ldots, \lambda_k$ of Σ.

Let $u_j \; for \; 1 \leq j \leq k$. Consider the appropriate eigenvalue of u_j,denoted λ_j, it holds that:

$$\Sigma u_j = \lambda_j u_j \rightarrow \frac{1}{n}\sum_{i=1}^{n} \emptyset(x_i)\emptyset(x_i)^T u_j = \lambda_j u_j \rightarrow \frac{1}{\lambda_j n}\sum_{i=1}^{n} \emptyset(x_i)\emptyset(x_i)^T u_j = u_j$$

Notice that $\emptyset(x_i): d' \; x \; 1, \emptyset(x_i)^T: 1 \; x \; d', u_j: d' \; x \; 1$ and thus $\emptyset(x_i)^T u_j: (1 \; x \; d')x(d' \; x \; 1) = 1 \; x \; 1$, meaning that $\emptyset(x_i)^T u_j$ is a scalar(which is compatible with the hint), so we get that:

$$u_j = \frac{1}{\lambda_j n}\sum_{i=1}^{n} \emptyset(x_i)^T u_j \emptyset(x_i) \rightarrow u_j = \sum_{i=1}^{n} \frac{\emptyset(x_i)^T u_j}{\lambda_j n} \emptyset(x_i)$$

Thus, we got that u_j is a linear combination of $\emptyset(x_1), \ldots, \emptyset(x_n)$ that is $u_j = \sum_{i=1}^{n} a_{j,i}\emptyset(x_i)$

where $a_{j,i} = \frac{\emptyset(x_i)^T u_j}{\lambda_j n}$. ■

(c) Use the above to show that the coefficients $\alpha_{j,i}$ can be calculated efficiently (without dependence in d')? (Hint: show that $\alpha_j = \frac{1}{\lambda_j}\Phi\mathbf{u}_j$ where Φ is the matrix whose rows are the $\phi(\mathbf{x}_i)$'s and λ_j is the eigenvalue of $\Phi^T\Phi$ corresponding to the principal component $\mathbf{u}_j$. Conclude that each vector of coefficients α_j is a an eigenvector of $\bar{K}$)

Solution:

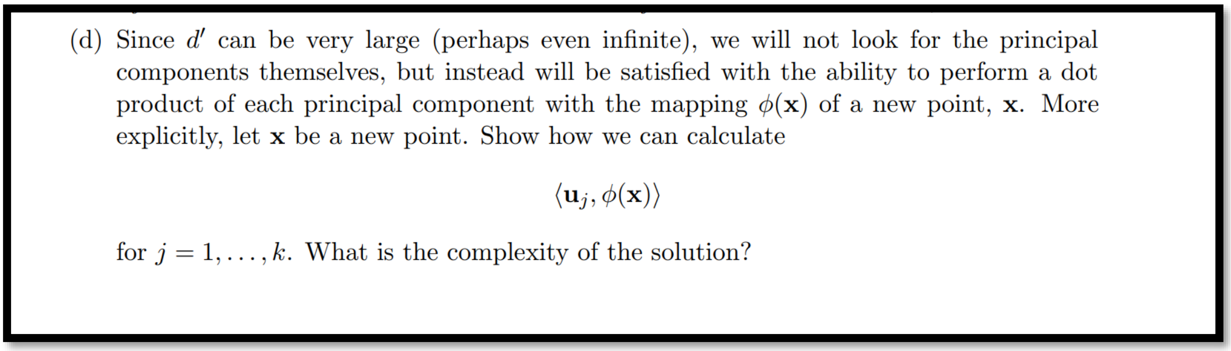
We showed in the question above that: $a_{j,i} = \frac{\emptyset(x_i)^T u_j}{\lambda_j n}$, and since $u_j = \sum_{t=1}^{n} a_{j,t}\emptyset(x_t)$ we get that:

$$a_{j,i} = \frac{\emptyset(x_i)^T u_j}{\lambda_j n} \rightarrow a_{j,i} = \frac{\emptyset(x_i)^T \sum_{t=1}^{n} a_{j,t}\emptyset(x_t)}{\lambda_j n} \rightarrow \lambda_j n a_{j,i} = \sum_{t=1}^{n} a_{j,t}\emptyset(x_i)^T\emptyset(x_t)$$

$$\rightarrow \lambda_j n a_{j,i} = \sum_{t=1}^{n} a_{j,t}\bar{K}_{i,t} = (\bar{K}{a_j}^T)_i$$

Meaning we got that $\lambda_j n a_{j,i} = (\bar{K}{a_j}^T)_i$, where $a_j = (a_{j,1}, \dots, a_{j,n})$. So we got that for every j the i-th entry(in the i-th row) in $\lambda_j n {a_j}^T\!: n\ x\ 1$ is equal to the i-th entry in $\bar{K}{a_j}^T\!: (n\ x\ n)\ x(n\ x\ 1) = n\ x\ 1$.

Thus, we get that $\lambda_j n {a_j}^T = \bar{K}{a_j}^T$ and thus we conclude that ${a_j}^T$ is an eigenvector of $\bar{K}$ with eigenvalue $\lambda_j n$, and thus we can calculate ${a_j}^T$(and thus we get the coefficients $a_{j,i}$) by finding the eigenvectors of $\bar{K}$ which is done without any dependence with d'. ■



(d) Since d' can be very large (perhaps even infinite), we will not look for the principal components themselves, but instead will be satisfied with the ability to perform a dot product of each principal component with the mapping $\phi(\mathbf{x})$ of a new point, $\mathbf{x}$. More explicitly, let $\mathbf{x}$ be a new point. Show how we can calculate

$$\langle \mathbf{u}_j, \phi(\mathbf{x}) \rangle$$

for $j = 1, \dots, k$. What is the complexity of the solution?

Solution:

Notice that according to the previous questions:

$$< u_j, \phi(x) > = < \sum_{i=1}^{n} a_{j,i}\emptyset(x_i) , \phi(x) > = \sum_{i=1}^{n} < a_{j,i}\emptyset(x_i), \phi(x) > = \sum_{i=1}^{n} a_{j,i} < \emptyset(x_i), \phi(x) >$$

$$= \sum_{i=1}^{n} a_{j,i}\, K(x_i, x)$$

Since we can calculate $K(x_i, x)$ in O(d) -the kernel trick- and we do this n times and after that sum the results we got so the time complexity is $O(nd)$.

So, we conclude that if we are satisfied with performing the dot product of each principal component with the mapping $\phi(x)$ of a new point, $< u_j, \phi(x) >$, without looking for the principal components itself we can do this in $O(nd)$.

■