CMU 10-715: Midterm Exam

DUE: Oct. 8, 2020, 11:40 AM Eastern time.

Instructions:

- It is a 24-hour exam. The submission site will close at 11:40 AM, Oct 8.
- You can refer to your own notes, the scribe notes, homework solutions, the text book, and lecture videos. You are **NOT** allowed to use any other resources (e.g., no searching on the Internet).
- You are **NOT** allowed to discuss the exam with anyone in the duration of the exam. For any questions, please post on Diderot.
- For the programming question(s), you can feel free to use any code you wrote for any of the homeworks.
- Submit your solutions in a pdf file to Gradescope. Handwritten or typed solutions are both accepted. Start **each question on a new** page and make sure you select the corresponding page(s) for each question during submission to Gradescope.
- We recommend taking a brief look at all questions first and then starting to answer them. You may find some questions easier than others. Good luck!

Distribution of Marks

Question	Points	Score
1	10	
2	10	
3	25	
4	10	
5	10	
6	15	
7	15	
8	15	
Total:	110	

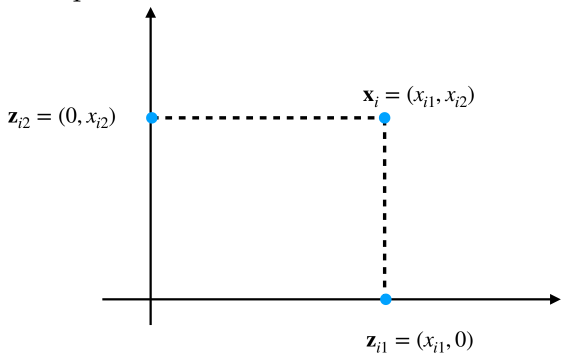
1) Enthusiastic Data Augmentation [10 pts]

This is a "conceptual" question. You don't need to give formal proofs. Giving a text-based description, examples and/or pictures is fine.

Consider binary classification, with $\mathcal{X} = \mathbb{R}^d$ for some $d \geq 2$ and $\mathcal{Y} = \{-1, 1\}$ and a training dataset $\{(\mathbf{x}_i, y_i)\}_{i \in [n]}$.

An enthusiastic student tries the following new approach towards training.

- The student creates a new training dataset of nd samples as follows.
  For each $i \in [n]$ and $j \in [d]$ define $\mathbf{z}_{ij} \in \mathbb{R}^d$ whose j^{th} coordinate equals the j^{th} coordinate of $\mathbf{x}_i$ and all other coordinates are zero. The following figure shows an example:

  
  $\mathbf{x}_i = (x_{i1}, x_{i2})$
  $\mathbf{z}_{i2} = (0, x_{i2})$
  $\mathbf{z}_{i1} = (x_{i1}, 0)$

  Now consider the modified training dataset: $\{(\mathbf{z}_{ij}, y_{ij})\}_{i \in [n]\ j \in [d]}$.
- The enthusiastic student trains a perceptron on this modified training set and obtains $\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$.
- Given a new point $\mathbf{x} \in \mathbb{R}^d$, the prediction is $f(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$.

What do you think about this idea? Are there settings where the perceptron algorithm can classify the original training data perfectly, but fails to do so on the modified training data?

2) Perceptron on Non Linearly Separable Data [10 pts]

Consider a binary classification problem, with $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, 1\}$ and a training dataset $\{(\mathbf{x}_i, y_i)\}_{i\in[n]}$. Consider the perceptron algorithm seen in class.

Algorithm 1: Perceptron algorithm

```
Initialize parameters $\mathbf{w}_0 = 0$, $b_0 = 0$, step $t = 0$;
while $\exists i \in [n]$ such that $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \leq 0$ do
    $\mathbf{w}_{t+1} = \mathbf{w}_t + y_i \mathbf{x}_i$;
    $b_{t+1} = b_t + y_i$;
    $t = t + 1$;
end
Output $\mathbf{w}_t$ and $b_t$
```

(a) (5 pts) With this context, what does it mean for a training dataset $\{(\mathbf{x}_i, y_i)\}_{i\in[n]}$ to not be linearly separable in terms of $\mathbf{w}$ and b? Please write down the formal mathematical meaning of it.

(b) (5 pts) Prove or give a counter example: The perceptron algorithm will never terminate when the training data is not linearly separable.

3) Kernel Soft SVM implementation [25 pts]

Consider the soft SVM objective:

$$\begin{aligned} \underset{\mathbf{w},\xi_i}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{n}\xi_i & \\ \text{subject to} \quad & y_i(\mathbf{w}^T\psi(x_i)+b) \geq 1-\xi_i & i=1,\dots,n \\ & \xi_i \geq 0 & i=1,\dots,n \end{aligned} \tag{1}$$

Equation 1 has its dual form:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \mathbf{1}^T\alpha - \frac{1}{2}\alpha^T\tilde{K}\alpha \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C \text{ for all } i, \quad \alpha^T y = 0 \end{aligned} \tag{2}$$

where α is the dual variable, $K_{ij} = \langle\psi(x_i),\psi(x_j)\rangle$, and $\tilde{K}_{ij} = y_i y_j K_{ij}$.

In this question you will implement your own Kernel Soft SVM and test it on the data data.txt[1] using the CVXOPT library `https://cvxopt.org/`.

You should use the Radial Basis Function kernel:

$$\langle\psi(x_i),\psi(x_j)\rangle = K_{ij} = \exp\left(-\gamma||x_i - x_j||_2^2\right)$$

You are not allowed to use packages that directly implement the kernel SVM algorithm. Please append your code to the end of your pdf submission. Your code will **NOT** be evaluated by the Autograder.

(a) (5 pts) The quadratic programming solver method from CVXOPT solvers.qp() uses as input the matrices $\mathbf{Q},\mathbf{p},\mathbf{G},\mathbf{h},\mathbf{A},\mathbf{b}$.[2] Write the correspondence between these inputs and the matrices of Equation 2.

(b) (20 pts) Fit the classifier with $\gamma = \{4, 100\}$ and $C = 1$ on the training data using CVXOPT. After you fit your classifier, plot the original data and the decision boundaries for $\gamma = \{4, 100\}$. You can use the python function contourf on a fine meshgrid to plot the decision boundary.

[1]The data is provided to you on this link `https://github.com/ShenghaoWu/10715/`

[2]Documentation: `https://cvxopt.org/userguide/coneprog.html#quadratic-programming`. The $\preccurlyeq$ symbol represents componentwise inequality.

4) Kernelizable Objectives and the Representer Theorem [10 pts]

Recall the Representer Theorem (and assume it as a given). Let $\psi : \mathcal{X} \to \mathcal{F}$ where $\mathcal{F}$ is Hilbert space. Consider $f : \mathbb{R}^n \to \mathbb{R}$. Now, there are many algorithms which optimize over the squared norm $||\mathbf{w}||^2$. Moreover, when trying to write down some algorithms in the form required by Mercer's theorem, they have some other function of $\mathbf{x_i}$'s as well in their objective. With this motivation, your goal is to prove that algorithms that optimize the following objective are also kernelizable:

$$\arg\min_{\mathbf{w}\in\mathcal{F}} \quad f(\langle \mathbf{w}, \psi(\mathbf{x_1})\rangle, ..., \langle \mathbf{w}, \psi(\mathbf{x_n})\rangle) + \tilde{R}(||\mathbf{w}||, ||\mathbf{w}||^2) + g(\mathbf{x_1}, ..., \mathbf{x_n}).$$

where $f(\langle \mathbf{w}, \psi(\mathbf{x_1})\rangle, ..., \langle \mathbf{w}, \psi(\mathbf{x_n})\rangle)$ and $g(\mathbf{x_1}, ..., \mathbf{x_n})$ are arbitrary functions, $\tilde{R}(||\mathbf{w}||, ||\mathbf{w}||^2)$ is coordinatewise non-decreasing.

5) Validity of a Kernel [10 pts]

Consider $\mathcal{X} = \mathbb{R}^d$ for some $d \geq 2$. Consider the function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as:

$$K(\mathbf{x}, \tilde{\mathbf{x}}) = \log\left(\frac{1}{d}\right), \text{ for all } \mathbf{x}, \tilde{\mathbf{x}} \in \mathcal{X}$$

Prove or disprove: K is a valid kernel.

6) Epsilon-net Arguments [15 pts]

Although the epsilon-net argument works in many other problems, there are flaws in the arguments below. Your goal is to identify any one of the five steps below which does NOT work and justify your answer. To be clear, you do not have to show whether other steps work or not – just need to identify one step and show that the claim in that step is false.

Consider the realizability and i.i.d. assummptions. In the lecture, we proved a bound on the sample complexity of PAC learning when the hypothesis class $\mathcal{H}$ is a finite-sized class. In particular, we showed that $\mathcal{H}$ is (ϵ, δ)-PAC learnable as long as the number of samples is at least $\left\lceil \frac{log(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil$. However, if the class $\mathcal{H}$ is infinite-sized, then however simple this class may be, simply substituting it in the bound above gives a vaccuous result.

A common trick for using such results for finite settings in order to get results for infinite settings is called 'epsilon-net arguments'. The high-level idea is to (i) quantize the infinite setting to obtain a finite setting, (ii) use the finite result, and (iii) show that the quantization doesn't hurt too much.

Let us try that approach here.
Consider $\mathcal{X} = [0, 1]$ and $\mathcal{H} = \{h(x) = \text{sign}(x - \beta) | \beta \in [0, 1]\}$. Clearly, $|\mathcal{H}| = \infty$. The goal is to use the above PAC learning result (which was derived for finite-sized $\mathcal{H}$) to obtain PAC learnability for this class via an epsilon-net argument.

1. For some (finite) positive integer k, define $\mathcal{H}_k$ as the hypotheses have thresholds $0, 1/k, 2/k, ..., 1$, that is, $\mathcal{H}_k = \{h(x) = \text{sign}(x - \beta) | \beta \in \{0, 1/k, 2/k, ..., 1\}\}$. Thus we have $\mathcal{H}_k \subseteq \mathcal{H}$.
2. Since $|\mathcal{H}_k| = k + 1 < \infty$, we can use the aforementioned result to get a sufficient sample size for $(\epsilon/2, \delta)$-PAC learnability of $\mathcal{H}_k$ as $\left\lceil \frac{2\log((k+1)/\delta)}{\epsilon} \right\rceil$.
3. We can then show that there exists a large enough finite value of k such that for every $h^* \in \mathcal{H}$, there exists some $h_k \in \mathcal{H}_k$ such that $R_{(D,h^*)}(h_k) \leq \epsilon/2$ for every distribution D. Denote this value of k as ξ.
4. Finally consider the Empirical Risk Minizer (ERM) on class $\mathcal{H}_\xi$ and denote the output as h_{ERM}. Consider any true classifier $h^* \in \mathcal{H}$ and any distribution D. From point 3 above, let $h_{approx} \in \mathcal{H}_\xi$ be a hypothesis such that $R_{(D,h^*)}(h_{approx}) \leq \epsilon/2$. Show that $R_{(D,h^*)}(h_{ERM}) \leq R_{(D,h^*)}(h_{approx}) + R_{(D,h_{approx})}(h_{ERM})$.
5. Conclude that $\mathcal{H}$ is (ϵ, δ)-PAC learnable if the number of samples is at least $\lceil \frac{2\log(|\mathcal{H}_\xi|/\delta)}{\epsilon} \rceil = \lceil \frac{2\log(|(\xi+1)|/\delta)}{\epsilon} \rceil$.

7) PAC Learnable [15 pts]

Let a classification problem be defined by $\mathcal{X} = \mathbb{R}^d$ for some $d \geq 1$ and $\mathcal{Y} = \{-1, 1\}$.
For any value of integer $m \geq 1$ and any $\mathbf{x_1}, \ldots, \mathbf{x_m} \in \mathcal{X}$, define the hypothesis class:

$$\mathcal{H}(\mathbf{x}_1, \ldots, \mathbf{x}_m) = \{h : \mathcal{X} \to \mathcal{Y} \mid h(\mathbf{x}_1) = \cdots = h(\mathbf{x}_m) = 1, \text{ and } h(\mathbf{x}) = -1 \text{ for other } \mathbf{x} \in \mathcal{X}\}.$$

Now define,

$$\mathcal{H}_m = \bigcup_{\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathcal{X}} \mathcal{H}(\mathbf{x}_1, \ldots, \mathbf{x}_m).$$

Finally define,

$$\mathcal{H} = \bigcup_{\text{all integers } m \geq 1} \mathcal{H}_m.$$

Is the class $\mathcal{H}$ PAC learnable (under realizability)? If yes, derive an upper bound on its sample complexity. Supplement your responses with proofs.

8) No Free Lunch Theorem [15 pts]

Consider any hypothesis class $\mathcal{H}$.

(a) (10 pts) Recall the no free lunch theorem: If the sample size n and the feature space $\mathcal{X}$ satisfy $n < \frac{|\mathcal{X}|}{2}$ then, for any learning algorithm (whose output is denoted by h) it must be that

$$\mathbb{P}(R(h) \geq \frac{1}{8}) \geq \frac{1}{7}.$$

Use the statement of the no free lunch theorem to prove that if the VC dimension of $\mathcal{H}$ is infinite, then $\mathcal{H}$ is not PAC learnable under realizability.

(b) (5 pts) Prove that if $\mathcal{H}$ is not PAC learnable under the realizability assumption then it is not agnostically PAC learnable.