COL 774: Machine Learning
Minor 2 Examination

Wednesday March 6, 2019

Notes:

- **Time: 3:55 pm to 5:05 pm. Total Questions: 5. Maximum Points: 24.**
- **This exam is closed book/notes. One A4 sized sheet of handwritten notes is allowed as announced on class mailing list.**
- **This question paper has printed material on both sides.**
- **The points for each question are mentioned at the end of the question**.
- **Some questions may be harder than others. Use your time wisely.**
- **Start each answer on a new page.**
- **Justify all your answers. Answers without justification may not get full points.**
- **We will use the notation as described in class unless otherwise stated.**

1. **Understanding Conditional Probability:** Consider three Boolean valued random variables X_1, X_2 and Y. Construct a distribution over these variables such that $(X_1 \not\perp X_2)$ but $(X_1 \perp X_2 \,|Y)$. You should carefully justify why your construction satisfies the desired properties. (**4 points**)

2. **Solving SVMs in Primal:** Consider the following SVM optimization problem:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i \tag{1a}$$

$$y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i; \quad \forall i \tag{1b}$$

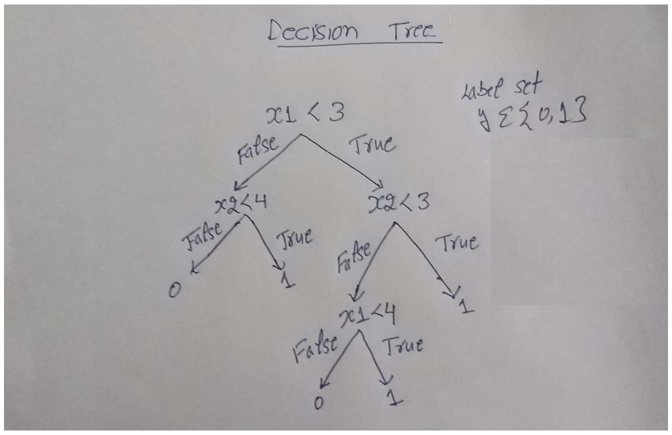
$$\xi_i \geq 0; \quad \forall i \tag{1c}$$

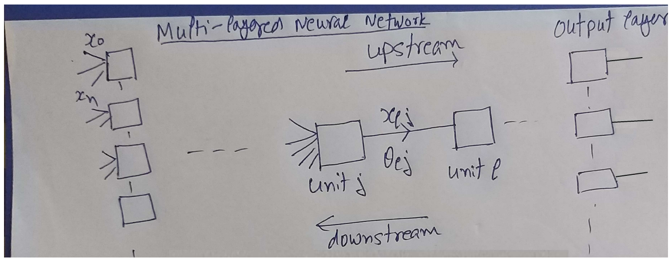
$$\tag{1d}$$

The symbols are as defined in class.

(a) Derive an equivalent unconstrained formulation for the above problem. Your derivation should be entirely in the primal space, i.e., you should not introduce any new variables such as Langrange multipliers. Clearly state at what points is your unconstrained objective non-differentiable. (**2 points**)

(b) Theory of sub-gradients generalizes gradients to optimize convex functions which are non-differentiable at a finite set of points. In its simplest form, at any given point, if the function is not differentiable, we substitute the value of either the left or the right gradient as a proxy for the gradient, and refer to it as a sub-gradient at that point. For example, if $f(x) = |x|$ $(x \in \mathcal{R})$, then at $x = 0$, we can define the sub-gradient to be 1 (right derivative) or -1 (left derivative).





(a) Decision Tree

(b) Multi-layered Neural Network

Figure 1: (a) Neural Network (b) Decision Tree

Clearly derive the gradient expression with respect to the variables w, b in your unconstrained SVM objective (part (a) above). Make use of sub-graidents wherever the function is non-differentiable. (**3 points**)

3. **Kernels:** Consider a kernel function defined as $K(x,z) = (x^Tz + c)^2$ where $x, z \in \mathcal{R}^n$ and $c \in \mathcal{R}$. Show that the corresponding kernel matrix K^M is positive semi-definite. You should prove it from first principles and not make use of Mercer's theorem. (**4 points**)

4. **Decision Trees:**

   (a) Consider the decision tree shown in Figure 1a. Here, assume $x_1, x_2 \in \mathcal{R}$, and label $y \in \{0,1\}$. Draw the decision boundary corresponding to this tree in $\mathcal{R}^2$. (**3 points**)

   (b) Next, consider learning a decision tree in $\mathcal{R}^n$ i.e., with n features given as, $x_1, x_2, \cdots x_n \in \mathcal{R}$. Let the label space be given by $y \in \{0,1\}$. You can assume that each internal node splits on conditions of the form $x_j \geq v$ or $x_j \leq v$, where x_j is some attribute and $v \in \mathcal{R}$. Describe the form of the decision boundary in terms of the hyperplanes in $\mathcal{R}^n$. Can the decision tree learner described above learn arbitrary hyperplanes? How are these hyperplanes being combined with each other to form the decision boundary? Justify your answer. (**2 points**)

5. **Weight Learning in Neural Networks:** Consider learning the parameters of a multi-layered neural network. For this question, we will refer to the diagram shown in Figure 1b. Here, unit j and unit l refer to some perceptrons in the neural network such that l is upstream neighbor of j. Here is some notation:

   - x_{lj} refers to the input from unit j to l.
   - θ_{lj} refers to the weight of connection from unit j to l.
   - net_j refers to the net linear input into unit j and is given as $net_j = \Sigma_{k \in downNbrs(j)} \theta_{kj} x_{kj}$.
   - o_j refers to the net output of unit j and is obtained after applying a sigmoid function over net_j, i.e., $o_j = \frac{1}{1+exp^{-net_j}}$.
   - We have $x_{lj} = o_j \ \forall l, l \in upNbrs(j)$. This is because all the upstream neighbors of j get the same input which is exactly the output of unit j.

- $J(\theta)$ refers to the net training loss incurred by the network for a given set of θ parameters in the network.

Next, answer the following questions.

(a) Use the (generalized) chain rule to express the value of $\frac{\partial J(\theta)}{\partial net_j}$ in terms of the $\frac{\partial J(\theta)}{\partial net_l}$'s where l denotes an upstream neighbor of j. The only other terms allowed in your expression are partial derivatives of net_j with respect to net_l's (l being upstream neighbor of j). (**2 points**)

(b) Next, compute the value of $\frac{\partial net_l}{\partial net_j}$ expressed in terms of o_j and θ_{lj}. (**2 points**)

(c) Assuming that you have access to $\frac{\partial J(\theta)}{\partial net_j}$ for the units in the output layer, and making use of above expressions (in parts (a) and (b)), write down an algorithm to compute $\frac{\partial J(\theta)}{\partial net_j}$ for units in each layer in turn starting from the output layer. (**2 points**)