HOMEWORK 7

Yichen Lin
908 531 9599

Instructions: Use this latex file as a template to develop your homework. Please submit a single pdf to Canvas. Late submissions may not be accepted. You can choose any programming language (i.e. python, R, or MATLAB). Please check Piazza for updates about the homework.

Note: All code and discussion about questions that require coding are attached in the .ipynb file

1 Kernel SVM [15 pts]

Consider the following kernel function defined over $z, z' \in Z$:

$$k(z, z') = \begin{cases} 1 & \text{if } z = z', \\ 0 & \text{otherwise.} \end{cases}$$

1. (5 pts) Prove that for any integer $m > 0$, any $z_1, \dots, z_m \in Z$, the $m \times m$ kernel matrix $K = [K_{ij}]$ is positive semi-definite, where $K_{ij} = k(z_i, z_j)$ for $i, j = \{1 \dots m\}$. (Let us assume that for $i \neq j$, we have $z_i \neq z_j$.)

   For $i \neq j$, we have $z_i \neq z_j$. Therefore, $K_ij = 0$, $K_ii = 1$ and K is an identity matrix. So K is positive semi-definite.

2. (5 pts) Given a training set $(z_1, y_1), \dots, (z_n, y_n)$ with binary labels, the dual SVM problem with the above kernel k will have parameters $\alpha_1, \dots, \alpha_n, b \in \mathbb{R}$. (Assume that for $i \neq j$, we have $z_i \neq z_j$.) The predictor for input z takes the form

$$f(z) = \sum_{i=1}^{n} \alpha_i y_i k(z_i, z) + b.$$

   Recall the label prediction is $\text{sgn}(f(z))$. Prove that there exists $\alpha_1, \dots, \alpha_n, b$ such that f correctly separates the training set. In other words, k induces a feature space rich enough such that in it any training set can be linearly separated.

3. (5 pts) How does that f predict input z that is not in the training set?

Comment: One useful property of kernel functions is that the input space Z does not need to be a vector space; in other words, z does not need to be a feature vector. For all we know, Z can be turkeys in the world. As long as we can compute $k(z, z')$, kernel SVM works on turkeys.

2 Game of Classifiers [50 pts]

2.1 Implementation

Implement the following models in choice of your programming language. Include slack variables in SVM implementation if needed. You can use autograd features of pytorch, tensorflow etc. or derive gradients on your own. (But don't use inbuilt models for SVM, Kernel SVM and Logistic Regression from libraries)

- Implement Linear SVM (without kernels).
- Implement Kernel SVM, with options for linear, rbf and polynomial kernels. You should keep the kernel parameters tunable (e.g. don't fix the degree of polynomial kernels but keep it as a variable and play with different values of it. Is Linear SVM a special case of Kernel SVMs?
- Implement Logistic Regression with and without kernels (use same kernel as Question 1).

2.2 Evaluation on Synthetic Dataset

2.2.1 Synthetic Dataset-1 (20 pts)

Generate 2-D dataset as following,
Let $\mu = 2.5$ and I_2 be the 2×2 identity matrix. Generate points for the positive and negative classes respectively from $\mathcal{N}([\mu, 0], I_2)$, and $\mathcal{N}([-\mu, 0], I_2)$. For each class generate 750 points, (1500 in total). Randomly create train, validation and test splits of size 1000, 250, 250 points respectively. Do the following with this dataset:

- (5 pts) Train your Linear SVM, Logistic Regression models and report decision boundaries, test accuracies.
- (5 pts) Show decision boundaries with K-NN and Naive Bayes Classifiers. (You can use library implementations or implement from scratch. Figure out the hyper-parameters using the validation set.)
- (5 pts) Repeat the process by varying μ from 1.0 to 2.4 with step size of 0.2, for each value of μ obtain test accuracies of the models and plot (μ on x-axis and test accuracy on y-axis). (You will have a curve for each of the 4-classifiers mentioned above)
- (5 pts) What are your conclusions from this exercise?

2.2.2 Synthetic Dataset-2 (20 pts)

Generate 1500 data points from the 2-D circles dataset of sklearn (`sklearn.datasets.make_circles`). Randomly create train, validation and test splits of size 1000, 250, 250 points respectively. Evaluate the above classifiers on this setting.

- (5 pts) Show decision boundaries for Linear SVM and Logistic Regression classifiers.
- (5 pts) Show decision boundaries for Kernel SVM and Kernel Logistic Regression (use rbf, polynomial kernels). Try different values of hyperparameters, report results with whichever works the best.
- (5 pts) Train Neural Network from HW4, and K-NN classifiers on this dataset and show decision boundaries. (You can use library implementation for these classifiers).
- (5 pts) What are your conclusions from this exercise?

2.3 Evaluation on Real Dataset (10 pts)

Lets put all this to some real use. For this problem use the the Wisconsin Breast Cancer dataset. You can download it from sklearn library(`sklearn.datasets.load_breast_cancer`)

- (5 pts) Do all the points of section 2.2.2 on this dataset. Since this is high-dimensional data, so you don't have to show the decision boundaries. Just report test accuracies for these classifiers and discuss your findings.
- (5 pts) In addition, you also want to figure out the important features which determine the class. Which regularization will you use for this? Upgrade your SVM, Kernel SVM implementation to include this regularization. Discuss what are the important features that you obtain by running your regularized SVM on this dataset. (You might have to normalize this dataset before training any classifier).

3 VC dimension [20 pts]

1. (7 pts) Let the input $x \in \mathcal{X} = \mathbb{R}$. Consider $\mathcal{F} = \{f(x) = \text{sgn}(ax^2 + bx + c) : a, b, c \in \mathbb{R}\}$, where $\text{sgn}(z) = 1$ if $z \geq 0$, and 0 otherwise. What is $VC(\mathcal{F})$? Prove it.

   Solution:

   (1) $a = b = c = 0$. Then $f(x) = 0$ for all x in x-axis, so $VC(\mathcal{F})$ is 1.

   (2) $a = b = 0$. Then $f(x) = sgn(c)$, which can be represented as a threshold c at the x-axis, so $VC(\mathcal{F})$ is 2.

   (3) $a = 0$. Then $f(x) = sgn(bx + c)$, which can be represented as a threshold $-\frac{c}{b}$ at the x-axis, so $VC(\mathcal{F})$ is also 2.

   (4) $a, b, c \neq 0$. Then $f(x) = sgn(ax^2 + bx + c)$, which now can be represented as two thresholds $-\frac{b+\sqrt{b^2-4ax}}{2a}$ and $-\frac{b-\sqrt{b^2-4ax}}{2a}$, so $VC(\mathcal{F})$ is 3

Shatter Coefficient and VC dimension

The main intuition behind VC theory is that, although a collection of classifiers may be infinite, using a finite set of training data to select a good rule effectively reduces the number of different classifiers we need to consider. We can measure the effective size of class $\mathcal{F}$ using *shatter coefficient*. Suppose we have a training set $D_n = \{(x_i, y_i)\}_{i=1}^n$ for a binary classification problem with labels $y_i = \{-1, +1\}$. Each classifier in $\mathcal{F}$ produces a binary label sequence

$$\big(f(x_1), \cdots, f(x_n)\big) \in \{-1, +1\}^n$$

There are at most 2^n distinct sequences, but often no all sequences can be generated by functions in $\mathcal{F}$. Shatter coefficient $\mathcal{S}(\mathcal{F}, n)$ is defined as the maximum number of labeling sequences the class $\mathcal{F}$ induces over n training points in the feature space $\mathcal{X}$. More formally,

$$\mathcal{S}(\mathcal{F}, n) = \max_{x_1, \cdots, x_n \in \mathcal{X}} \left| \Big\{ \big(f(x_1), \cdots, f(x_n)\big) \Big\} \in \{-1, +1\}^n, f \in \mathcal{F} \right|$$

where $|\cdot|$ denotes the number of elements in the set. The *Vapnik-Chervonenkis (VC) dimension* $V(\mathcal{F})$ of a class $\mathcal{F}$ is defined as the largest integer k such that $\mathcal{S}(\mathcal{F}, k) = 2^k$.

2. (7 pts) Suppose there are n points $(x_1, \cdots, x_n) \in \mathbb{R}$. Let $\mathcal{F}$ be the collection of 1-d linear classifiers: for each $t \in [0, 1]$, $f_t \in \mathcal{F}$ labels $x \leq t$ as -1 and $x > t$ as $+1$, or vice-versa. What is the shatter coefficient $S(\mathcal{F}, n)$? What is VC-dimension $V(\mathcal{F})$? How can you get it from shatter coefficient?

   For 1-d linear classifier, assuming that there are three points $x1 = -1, x2 = 0, x3 = 1$, and their labels are (1,-1,1), then the classifier can't shatter those points. And for two points, 1-d linear classifier can always shatter them

3. (6 pts) Show the following monotonicity property of VC-dimension: if $\mathcal{F}$ and $\mathcal{F}'$ are hypothesis classes with $\mathcal{F} \subset \mathcal{F}'$, then the VC-dimension of $\mathcal{F}'$ is greater than or equal to that of $\mathcal{F}$.

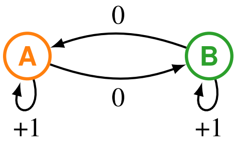
   If $\mathcal{F}$ and $\mathcal{F}'$ are hypothesis classes with $\mathcal{F} \subset \mathcal{F}'$, and the VC-dimension of $\mathcal{F}'$ is less than that of $\mathcal{F}$.

   Let S be a set of size k that can be shattered by $\mathcal{F}$. This means that for any labeling of the elements of S, there exists a hypothesis $h \in \mathcal{F}$ that separates the points in S. Since $\mathcal{F} \subseteq \mathcal{F}'$, we can conclude that there also exists a hypothesis $h \in \mathcal{F}'$ that separates the points in S. Therefore, S can be shattered by $\mathcal{F}'$ as well.

   So we get a contradictory conclusion. And therefore, if $\mathcal{F}$ and $\mathcal{F}'$ are hypothesis classes with $\mathcal{F} \subset \mathcal{F}'$, then the VC-dimension of $\mathcal{F}'$ is greater than or equal to that of $\mathcal{F}$.

4 Q-learning [15 pts]

Consider the following Markov Decision Process. It has two states s. It has two actions a: move and stay. The state transition is deterministic: "move" moves to the other state, while "stay' stays at the current state. The reward r is 0 for move, 1 for stay. There is a discounting factor $\gamma = 0.8$.



The reinforcement learning agent performs Q-learning. Recall the Q table has entries $Q(s, a)$. The Q table is initialized with all zeros. The agent starts in state $s_1 = A$. In any state s_t, the agent chooses the action a_t according to a behavior policy $a_t = \pi_B(s_t)$. Upon experiencing the next state and reward s_{t+1}, r_t the update is:

$$Q(s_t, a_t) \Leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha \left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') \right).$$

Let the step size parameter $\alpha = 0.5$.

1. (5 pts) Run Q-learning for 200 steps with a deterministic greedy behavior policy: at each state s_t use the best action $a_t \in \arg\max_a Q(s_t, a)$ indicated by the current Q table. If there is a tie, prefer move. Show the Q table at the end.

2. (5 pts) Reset and repeat the above, but with an ϵ-greedy behavior policy: at each state s_t, with probability $1-\epsilon$ choose what the current Q table says is the best action: $\arg\max_a Q(s_t, a)$; Break ties arbitrarily. Otherwise, (with probability ϵ) uniformly chooses between move and stay (move or stay both with 1/2 probability). Use $\epsilon = 0.5$.

3. (5 pts) Without doing simulation, use Bellman equation to derive the true Q table induced by the MDP.

   We can derive the following Bellman equations according to to our condition:

$$
\begin{aligned}
Q(A,S) &= 0.5Q(A,S) + 0.5(1 + 0.8Q(A,S)) \\
Q(A,M) &= 0.5Q(A,M) + 0.5(0 + 0.8Q(B,S)) \\
Q(B,S) &= 0.5Q(A,S) + 0.5(1 + 0.8Q(B,S)) \\
Q(B,M) &= 0.5Q(A,M) + 0.5(0 + 0.8Q(A,S))
\end{aligned}
$$

   So $Q(A,S) = 5, Q(A,M) = 4, (B,S) = 5, Q(B,M) = 4$