HOMEWORK 7

Anudeep Kumar
9084607069

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.

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 $z_1, \ldots, z_m \in Z$, the $m \times m$ kernel matrix $K = [K_{ij}]$ is positive semi-definite, where $K_{ij} = k(z_i, z_j)$ for $i, j = \{1 \ldots m\}$. (Let us assume that for $i \neq j$, we have $z_i \neq z_j$.)
   Given the condition, our Kernel matrix comes out to be

$$K = \begin{bmatrix} 1 & 0 & 0 & 0 & 0.. \\ 0 & 1 & 0 & 0 & 0.. \\ 0 & 0 & 1 & 0 & 0.. \\ 0 & 0 & 0 & 1 & 0.. \\ 0 & 0 & 0 & 0 & 1.. \\ . & & & & \\ . & & & & \end{bmatrix}$$

   This is nothing but a Identity matrix $I_{m \times m}$.
   To show that our Kernel matrix is positive semidefinte let us assume a $v \in \mathbf{R}^m \ni \forall v$ we have $v^T K v \geq 0$
   Now for $K = I_{m \times m}$ we get

$$v^T I v = \sum_{i=1}^{m} v_i^2 \geq 0$$

   Hence we can conclude our Kernel is positive semidefinite.

2. (5 pts) Given a training set $(z_1, y_1), \ldots, (z_n, y_n)$ with binary labels, the dual SVM problem with the above kernel k will have parameters $\alpha_1, \ldots, \alpha_n, b \in \mathbb{R}$. (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 $\alpha_1, \ldots, \alpha_n, b$ 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.
   As per the given condition

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

   We know for $k(z_i, z) if z_i = z \implies k = 1$
   So for the training set we have various values of z and y can have values -1 or +1. So looking at the dual

objective we can conclude that $f(z)$ can be written as

$$f(z) = \sum_{i \forall y_i = 1} \alpha_i y_i + \sum_{i \forall y_i = -1} \alpha_i y_i + b.$$

Since prediction is $sgn(f(z))$ we can write the prediction as

$$f(z) = \sum_{+ve} \alpha_i y_i - \sum_{-vevalue} \alpha_i y_i + b.$$

$$f(z) = (\sum_{+ve} \alpha_i - \sum_{-vevalue} \alpha_i) y_i + b.$$

Hence we can observe the above is linearly separable.

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

We know that

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

(a) Calculate $k(z_i, z)$ for each z_i.

(b) Calculate $\sum_{i=1}^{n} \alpha_i y_i k(z_i, z) + b$ for all $i = 1 \ldots n$

(c) If $(\sum_{i=1}^{n} \alpha_i y_i k(z_i, z) + b) \geq 0$ assign it to +1 class and -1 if otherwise. If $\sum_{i=1}^{n} \alpha_i y_i k(z_i, z) = 0$ then the prediction is $sgn(b), if b > 0$ class 1 and -1 otherwise.

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)

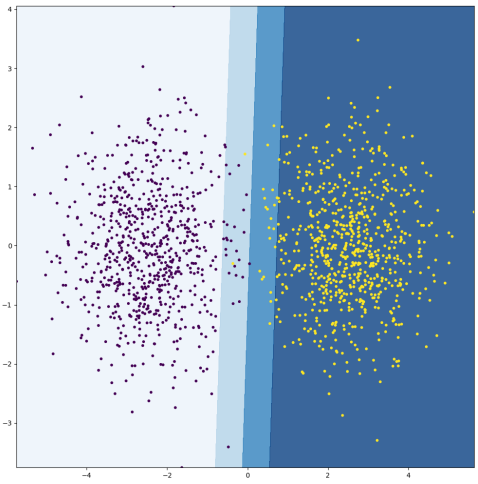
- Implement Linear SVM (without kernels).
- Implement Kernel SVM, with options for linear, rbf and polynomial kernels. You should keep the kernel parameters tunable (e.g. don't fix the degree of polynomial kernels but keep it as a variable and play with different values of it. Is Linear SVM a special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

2.2 Evaluation on Synthetic Dataset

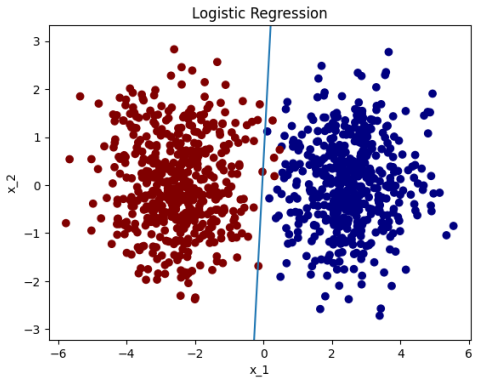
2.2.1 Synthetic Dataset-1 (20 pts)

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.
  Accuracy for Linear SVM : 99.2%
  Accuracy for Logistic Regression : 98.4%



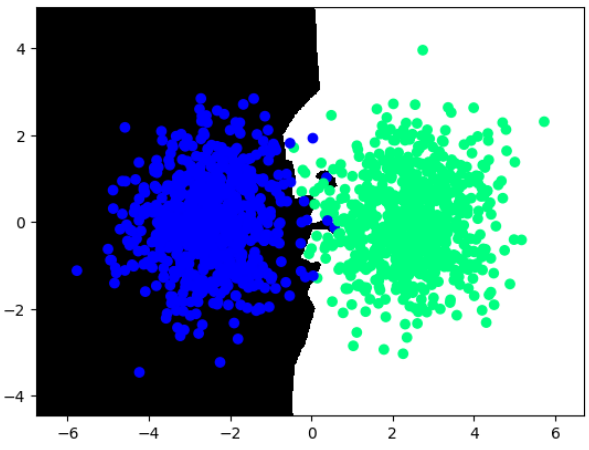
(a) Linear SVM



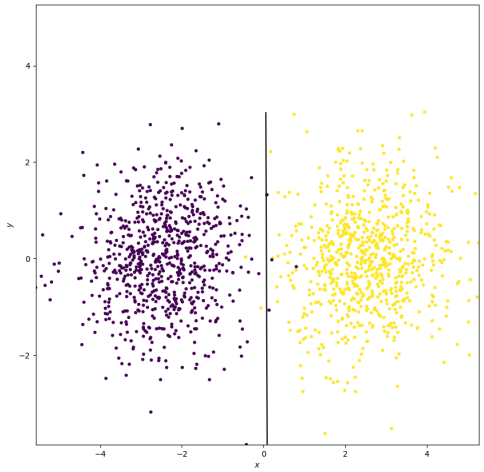
(b) Logistic Regression

Figure 1: Multivariate boundaries for Linear and Logistic Regression

- (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.)
  Accuracy for KNN : 98.399%
  Best K : 1
  Accuracy for Naive Bayes : 99.2%
  HyperParameter var smoothing which is portion of the largest variance of all features that is added to variances for calculation stability.
  varSmoothing: 0.533669923120631



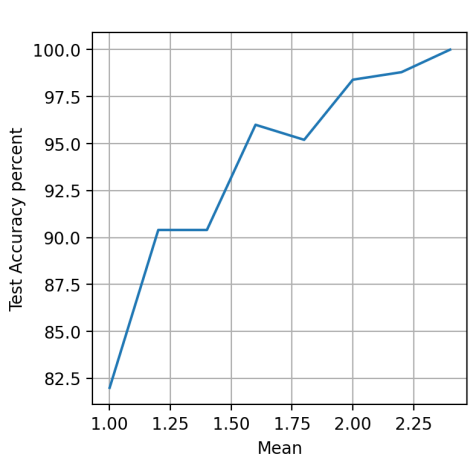
(a) KNN Classifier



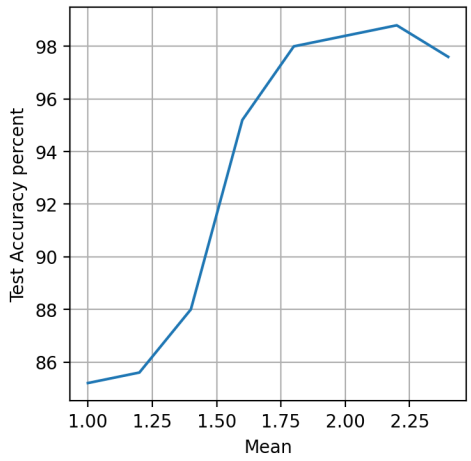
(b) Naive Bayes Classifier

Figure 2: Multivariate boundaries for KNN and Naive Bayes Classifier

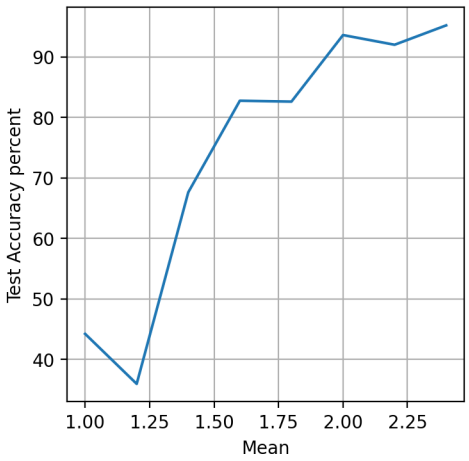
- (5 pts) Repeat the process by varying μ from 1.0 to 2.4 with step size of 0.2, for each value of μ obtain test accuracies of the models and plot (μ on x-axis and test accuracy on y-axis). (You will have a curve for each of the 4-classifiers mentioned above)



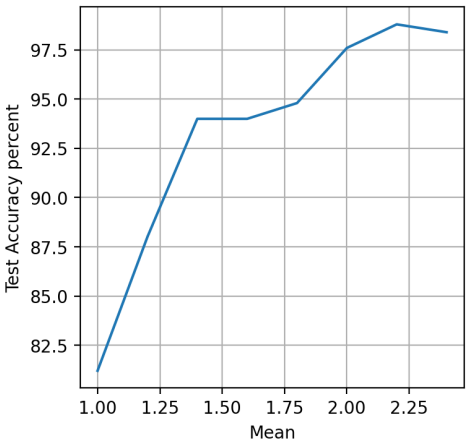
(a) Linear Regression



(b) Logistic Regression



(c) KNN Classifier



(d) Naive Bayes Classifier

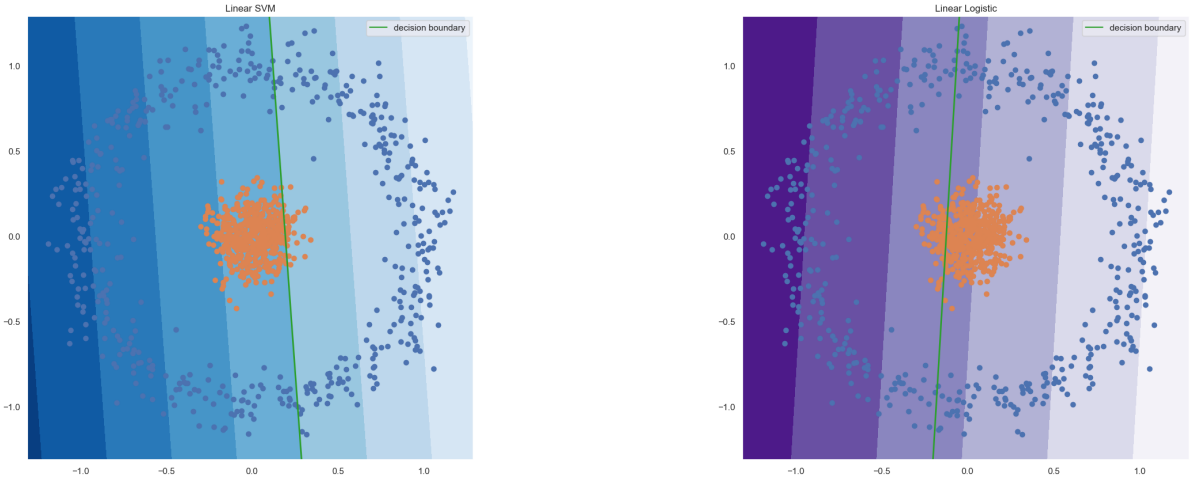
Figure 3: μ vs Accuaracy for various Classifiers

- (5 pts) What are your conclusions from this exercise?
  From the graphs above we can observe that as μ increases the accuracy of each classifier increases.This could be attributed to the fact that for smaller mean the data is very closely bound and hence it is difficult to linearly separate them but as mean increases it becomes easier to separate them.

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.
  Decision Boundary is in green
  Accuracy for Linear SVM : 68.8%
  Accuracy for Logistic Regression : 34.8%

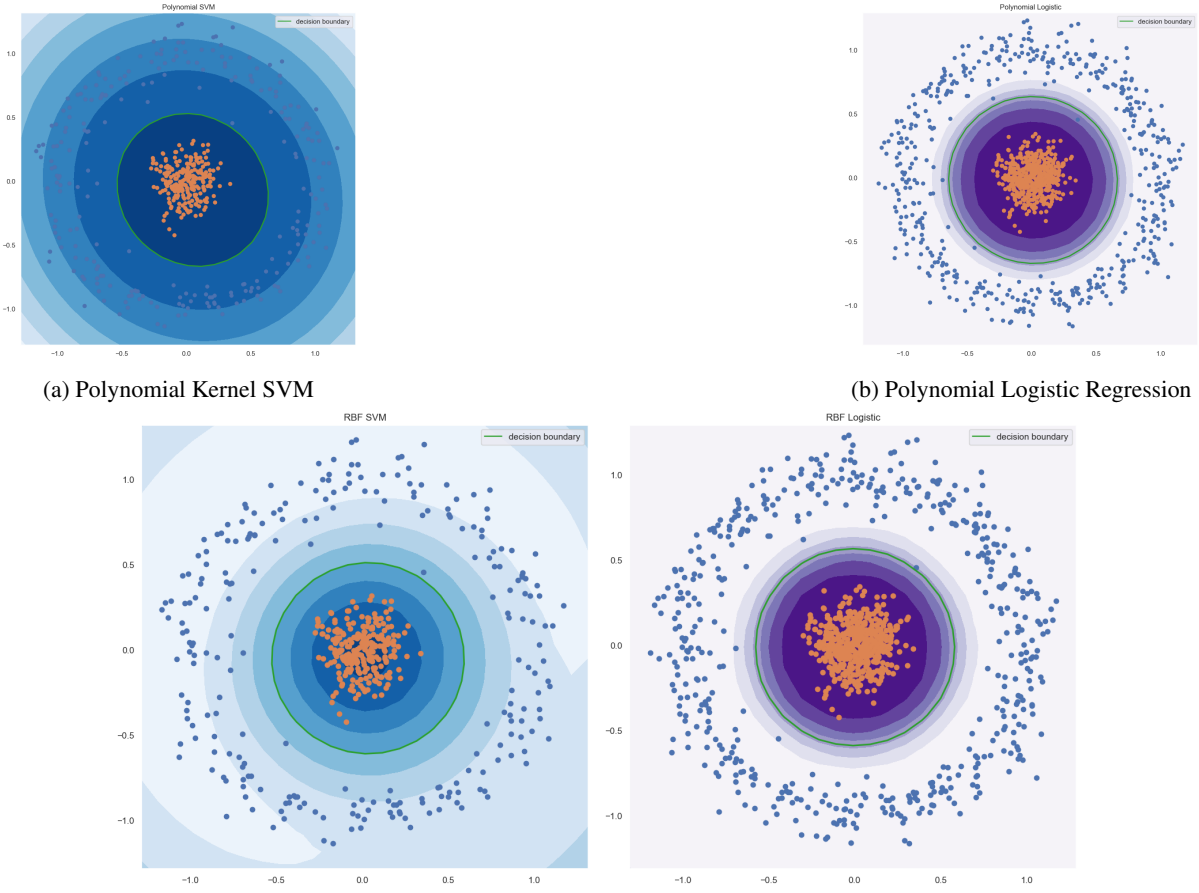


(a) Linear SVM (b) Logistic Regression

Figure 4: Make Circle Dataset decision boundary for Linear and Logistic Regression

- (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.

  Polynomial SVM with Degree =2 learning rate 0.01 Epochs = 1000 and additive bias c=1
  Polynomial Logistic Regression with Degree =2 learning rate 0.01 Epochs = 1000 and additive bias c=1

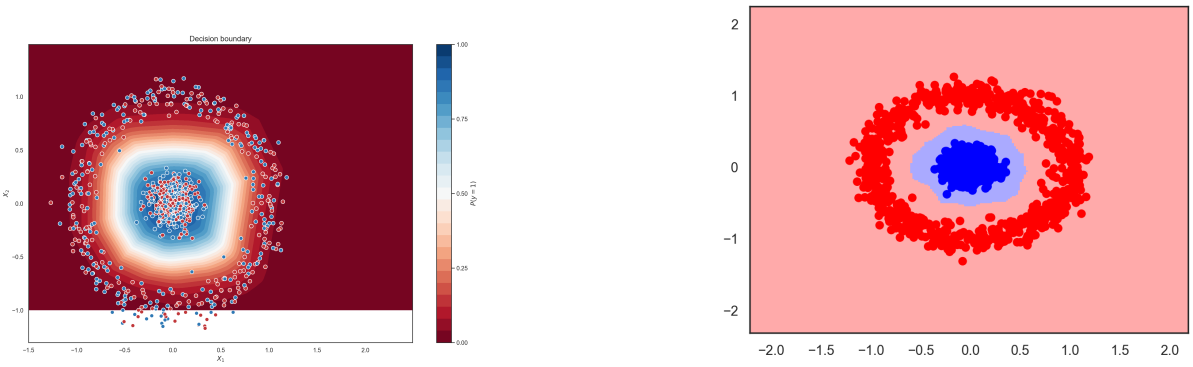


(a) Polynomial Kernel SVM (b) Polynomial Logistic Regression

(c) RBF SVM (d) RBF Logistic Regression

Figure 5: Decision boundary for Polynomial and RBF Kernels

- (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).
  Neural Network Hyperparameter:
  Learning Rate =0.05
  Epoch = 50
  Nuerons with 2 hidden Layers: $2 \times 30 \times 20 \times 1$



(a) Nueral Network on Make Circle

(b) KNN Classifier

Figure 6: Make Circle Dataset decision boundary for Nueral Network and KNN

- (5 pts) What are your conclusions from this exercise?
  We observe that for this dataset we have higher dimension data which cannot be linearly classified hence we observe how poorly Linear SVM and Logistic Regression performed.
  On the other hand Polynomial and RBF Kernel performed perfectly since they can capture higher dimensions easily.
  A small Neural Network with 2 hidden layer with 30 and 20 neurons performed perfectly. So did KNN with 1 Nearest Neighbour as best K.

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.
  For SVM Polynomial we set degree = 4 and epochs =10000
  For Logistic Polynomial degree =5 , c=1 , epochs =1000 ad learning rate =0.01

Accuracy(%)	Linear	Polynomial	RBF
SVM	60.66	90.66	90.66
Logistic	62	62	62

  KNN:
  K=1
  Accuracy :72.72%
  Neural Network :
  4 Hidden Layer with 30 neuron in input layer and 1 output : $3 \times 300 \times 200 \times 100 \times 10 \times 1$
  Accuracy : 96%

- (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).

We use L1 regularization since we need a sparse solution to narrow down the features we believe are important. After keeping $\lambda = 0.01$ in Linear SVM we obtain the following features as the ones having most weight : ['mean perimeter', 'mean area', 'worst perimeter']

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.

   As per the defintion of $\mathcal{F}$ we are dealing wit points on the x-axis. We can easily observe that for $VC(\mathcal{F}) \geq 2$ all combinations of 2^2 points can be shattered since a parabola can easily seperate combinations of $--, -+, +-, ++$. Now considering all 2^3 cases for the case of 3 points we can observe below :

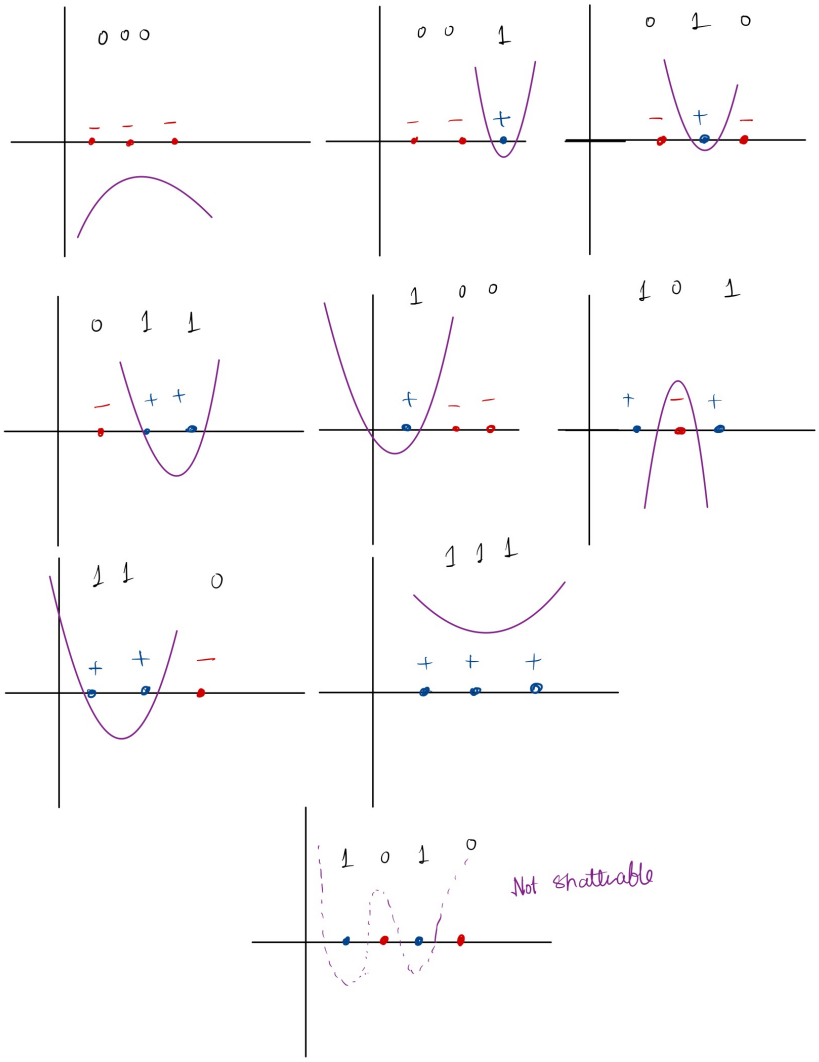


Figure 7: Eight shattered $VC(\mathcal{F})$ for 3 points and one non shatterable 4 point

Hence we can see above that $VC(\mathcal{F}) \geq 4$ cannot be shattered since the graph does not follow the given equation.

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?

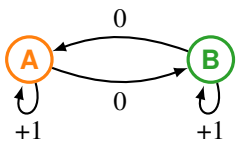
   Our given $\mathcal{F}$ deals with points on the x axis. As per the definition of shatter coefficient it is the maximum number of classifiers in $\mathcal{F}$ that are induced over n training points in feature space $\mathcal{X}$. For our given definition, when $x \leq t$ prediction is -1 and +1 for $x \geq 1$, we can imagine an axis with 1 point on it. This point can take 2 values, -1 and +1. For each of them we can find a threshold t such that we classify the point correctly always. If the point is -1, we find a t on the right of the point to classify it correctly. If the point is positive we can find a threshold to the left of it such that $x \geq t$ classifies correctly.Hence for 1 point we have 2 possible subsets and $\mathcal{F}$ induces 2 labeling sequences
   For a set of 2 points however, if we keep both the points ++ , – and -+ we can classify them correctly but if the points are +-, then we cannot find a t such that $x \leq t$ gives +1 and $x \geq t$ gives -1. Hence we have 3 subsets induced in case of 2 set of points which can have maximum of 4 subsets. So we can think of it as number of ways in which we can keep 1 threshold point between n different points. Which comes out to be $S(\mathcal{F}, n) = n + 1$.
   Now as per the definition of VC dimension, it is the max integer k for which Shatter Coefficient is 2^k. Using the value we obtained above, $k + 1 = 2^k$ which is solved for $k = 1$. Hence $V(\mathcal{F}) = 1$.

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 $\mathcal{F}$ is a subset of $\mathcal{F}'$ then all the labelling sequences that are induced by class $\mathcal{F}$ are a part of $\mathcal{F}'$. Hence the shattered subsets of $\mathcal{F}'$ will be at least equal to that of $\mathcal{F}$. If it contains more it will have a VC dimension more than $\mathcal{F}$.So the $V(\mathcal{F}') \geq V(\mathcal{F})$

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.

Q Value	Move	Stay
A	0	0
B	0	0

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 $\epsilon = 0.5$.

Q Value	Move	Stay
A	3.9971709460563005	4.997180395633022
B	3.994385161916294	4.998335051817343

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

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

Where

$$s_t = A, B$$

$$a_t = Move, Stay$$

Initially we have all all Q values 0 Given we start at A always, say we decide to Stay so
$s_1 = A$, $a_1 = Stay$,$s_{t+1} = A$ and $r_t = 1$

$$Q(s_1, a_1) \Leftarrow (1-\alpha)Q(s_1, a_1) + \alpha\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a')\right).$$

$$Q(s_1, a_1) \Leftarrow (1-0.5)0 + 0.5\,(1 + 0.8 \times 0)) = 0.5$$

So $Q(A, Stay) = 0.5$.
Next we are still on A and we decide to Move so, $s_2 = A$, $a_2 = Move$,$s_{t+1} = B$ and $r_2 = 0$

$$Q(s_2, a_2) \Leftarrow (1-0.5)0 + 0.5\,(0 + 0.8 \times 0)) = 0.0$$

So $Q(A, Move) = 0.0$
Next we are on B since we moved and we decide to stay so, $s_3 = B$, $a_3 = Stay$,$s_{t+1} = B$ and $r_3 = 1$

$$Q(s_3, a_3) \Leftarrow (1-0.5)0 + 0.5\,(1 + 0.8 \times 0.0)) = 0.5$$

So $Q(B, Stay) = 0.5$
Next we are on B since we stayed and we decide to move so, $s_4 = B$, $a_4 = Move$,$s_{t+1} = A$ and $r_4 = 0$

$$Q(s_4, a_4) \Leftarrow (1-0.5)0 + 0.5\,(0 + 0.8 \times 0.5)) = 0.2$$

So $Q(B, Move) = 0.2$
Next we are on A since we moved and say we decied to stay so, $s_5 = A$, $a_5 = Stay$,$s_{t+1} = A$ and $r_5 = 1$

$$Q(s_5, a_5) \Leftarrow (1-0.5)0.5 + 0.5\,(1 + 0.8 \times 0.5)) = 0.95$$

So $Q(A, Stay) = 0.95$
Next we are on A since we moved and say we decied to stay so, $s_5 = A$, $a_5 = Stay$,$s_{t+1} = A$ and $r_5 = 1$

$$Q(s_5, a_5) \Leftarrow (1 - 0.5)0.5 + 0.5\,(1 + 0.8 \times 0.5)) = 0.75$$

Next we are still on A and we decide to Move so, $s_6 = A$, $a_6 = Move$,$s_{t+1} = B$ and $r_6 = 0$

$$Q(s_6, a_6) \Leftarrow (1 - 0.5)0 + 0.5\,(0 + 0.8 \times 0.5)) = 0.2$$

So $Q(A, Move) = 0.2$
Next we are on B since we moved and we decide to stay so, $s_7 = B$, $a_7 = Stay$,$s_{t+1} = B$ and $r_7 = 1$

$$Q(s_7, a_7) \Leftarrow (1 - 0.5)0.5 + 0.5\,(1 + 0.8 \times 0.5)) = 0.95$$

So $Q(B, Stay) = 0.95$
Next we are on B since we stayed and we decide to stay so, $s_8 = B$, $a_8 = Stay$,$s_{t+1} = B$ and $r_8 = 1$

$$Q(s_8, a_8) \Leftarrow (1 - 0.5)0.95 + 0.5\,(1 + 0.8 \times 0.95)) = 0.95$$

So $Q(B, Stay) = 1.355$
Next we are on B since we stayed and we decide to stay so, $s_9 = B$, $a_9 = Stay$,$s_{t+1} = B$ and $r_9 = 1$

$$Q(s_8, a_8) \Leftarrow (1 - 0.5)1.355 + 0.5\,(1 + 0.8 \times 1.355)) = 0.1.7195$$

Similarly after 200 iteration we converge to values as induced by simulation.