HOMEWORK 7

APOORVA KUMAR
908 461 5997

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 $z_1, \dots, z_m \in Z$, 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 $z_i \neq z_j$.)

   The Kernal Gram Matrix with the condition $z_i \neq z_j$ for $i \neq j$ is an $m \times m$ Indentity Matrix:

   $$\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

   To show that our Kernel matrix is positive semidefinte if $\forall v \in \mathbb{R}^m$ 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 $(z_1, y_1), \dots, (z_n, y_n)$ 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.

   For any element (z_i, y_i) in the training data where $y_i \in \{1, -1\}$:

   $$\begin{aligned} f(z) &= \sum_{i=1}^{n} \alpha_i y_i k(z_i, z) + b \\ &= \alpha_i y_i + b \\ \text{sgn}(f(z)) &= \text{sgn}(\alpha_i y_i + b) \\ \hat{y_i} &= \begin{cases} \alpha_i + b & y_i = 1 \\ -\alpha_i + b & y_i = -1 \end{cases} \end{aligned}$$

Thus setting $\alpha_i > b$ for $y_i = 1$ and $\alpha_i < b$ for $y_i = -1$ we can perfectly separate the data because α_i doesn't affect any other data point for $i \neq j$.

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

For any $z \notin (z_i)_{i=1}^n$

$$k(z, z_i) = 0 \; \forall \, i = \{1, 2, ..., n\}$$

Thus,

$$\text{sgn}(f(z)) = \text{sgn}(b)$$

So, we can see that for any z not in training set have $\hat{y} = \text{sgn}(b)$.

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?

  Yes, Linear SVM is a special case of Kernel SVM.
- Implement Logistic Regression with and without kernels (use same kernel as Question 1).

2.2 Evaluation on Synthetic Dataset

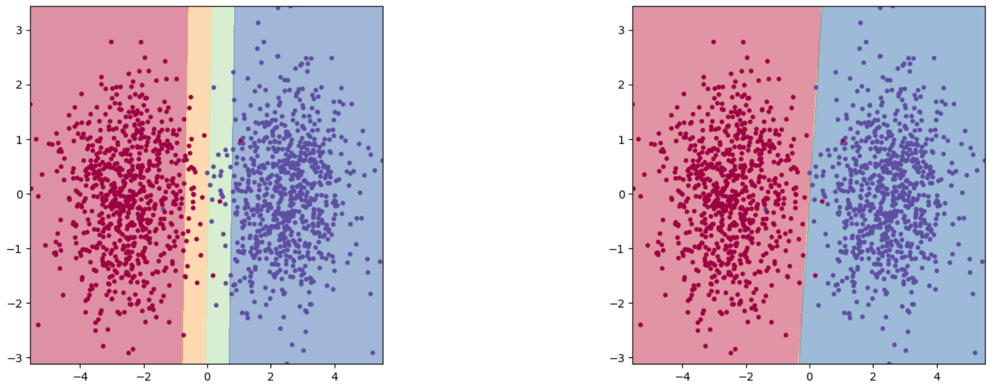
2.2.1 Synthetic Dataset-1 (20 pts)

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.

  SVM Test Accuracy: 99.6%

  Logistic Regression Test Accuracy: 99.6%



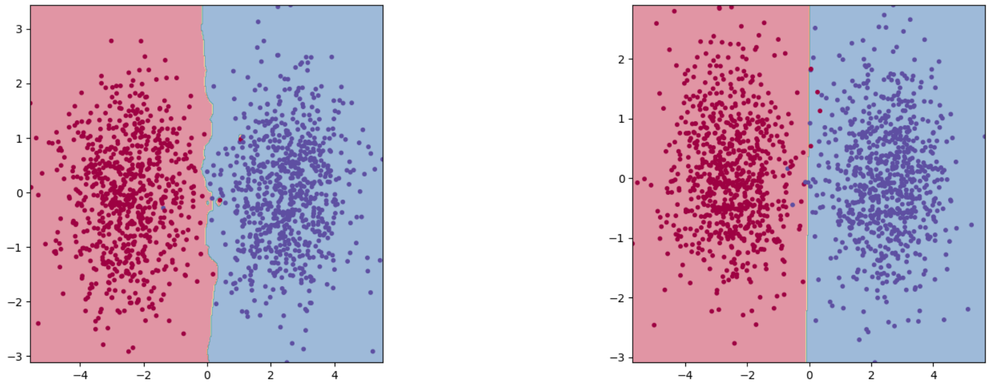
(a) SVM

(b) Logistic Regression

Figure 1: Decision Boundaries

- (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.)

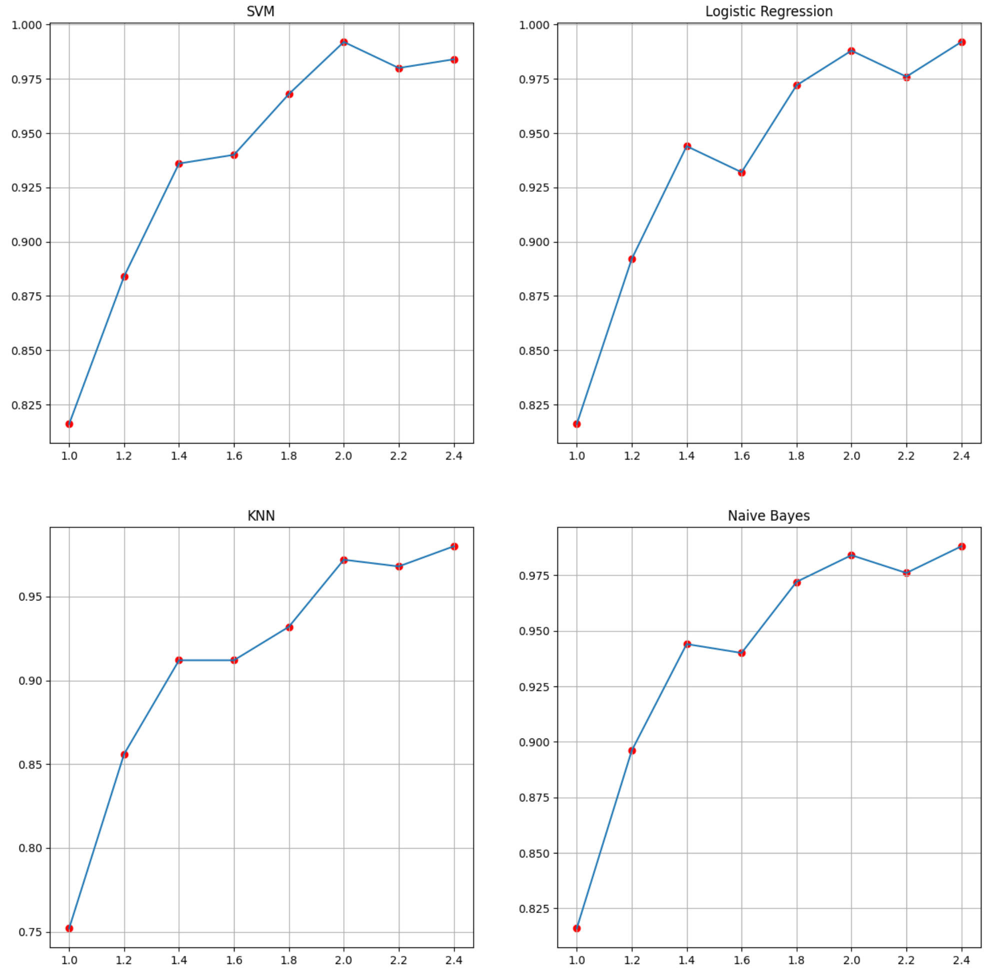
  KNN Test Accuracy for **K=3** = 100%
  Naive Bayes Test Accuracy for smoothing parameter as 1 = 99.2%



(a) KNN (b) Naive Bayes

Figure 2: Decision Boundaries

- (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?

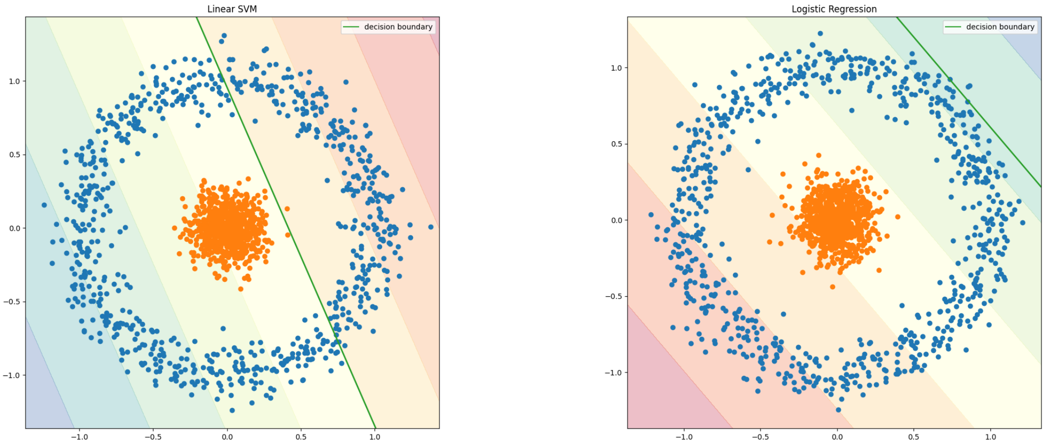
  As μ increase the data becomes more seperable and it becomes much easier to classifier the data. We also see that all of them form almost a linear decision boundary even including the KNN classifier.

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.

  Linear SVM has 60.33 % and Logistic has 32.33 % test accuracy.



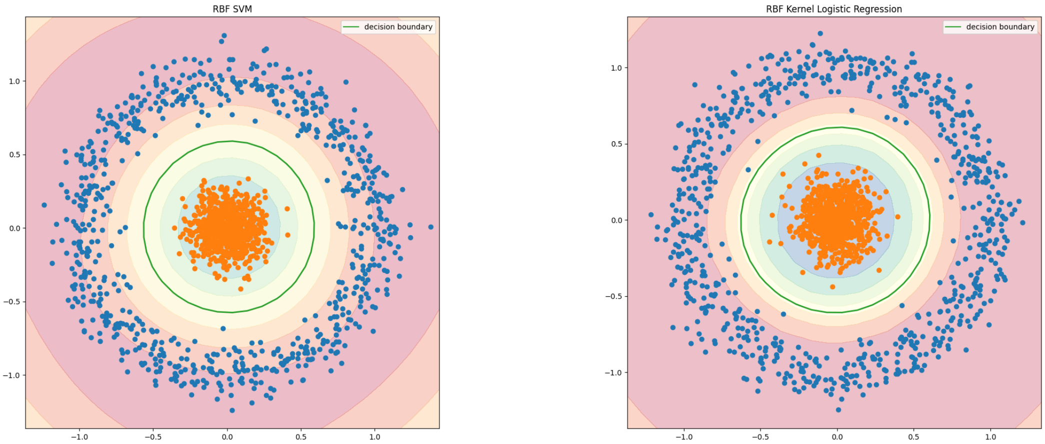
(a) Linear SVM

(b) Logistic Regression

Figure 3: Decision Boundaries

- (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.

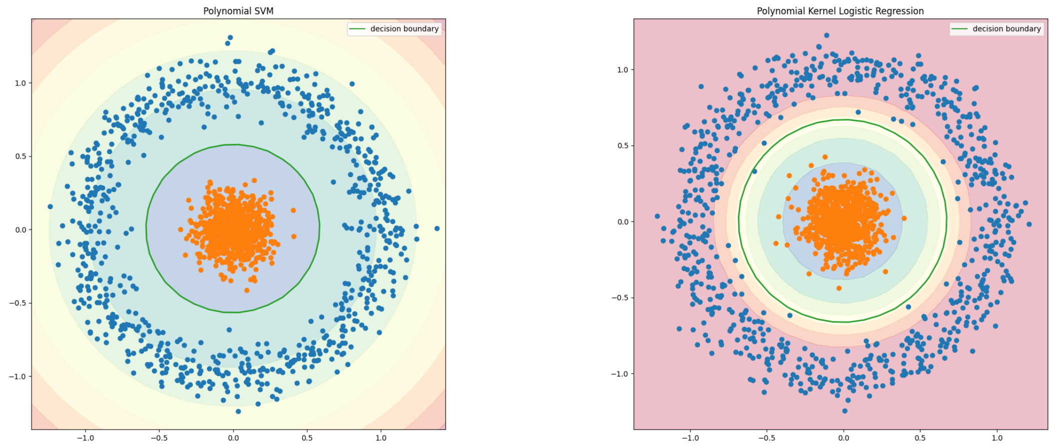
  All of these have 100% test accuracy.



(a) SVM

(b) Logistic Regression

Figure 4: Decision Boundaries for RBF Kernel

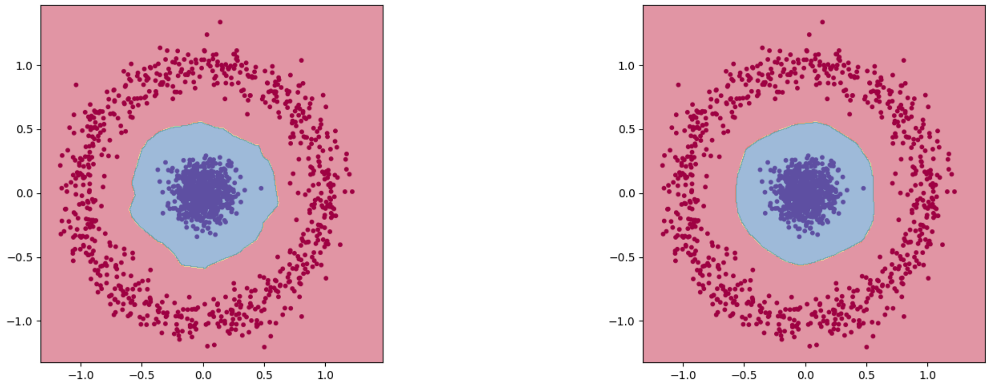


(a) SVM (b) Logistic Regression

Figure 5: Decision Boundaries for Polynomial Kernel

- (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).

  Both have 100% test accuracy



(a) KNN (b) Neural Network

Figure 6: Decision Boundaries

- (5 pts) What are your conclusions from this exercise?

  We see that for a dataset which is not linearly separable, kernels help a lot to ease the process by projecting the dataset in a space where it can be easily classified. This eases our process a lot wherein we can just plug and play a kernel into the process.

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.

  Data was split into 369 samples for training, 100 for validation and 100 for testing.

Model Name	Test Accuracy
Linear SVM	62%
RBF SVM	89%
Polynomial SVM (p=5)	76%
Logistic Regression	62%
RBF Logistic Regression	62%
Polynomial Logistic Regression (p=10)	62%
KNN(k=6)	90%
Neural Network	92%

- (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).

  Test Losses for Regularized SVM:

  1. Linear = 62 %
  2. Polynomial = 78 %
  3. RBF = 91 %

  The important features chosen were: ['mean perimeter', 'mean area', 'mean smoothness', 'mean concavity', 'texture error', 'concavity error', 'concave points error', 'worst perimeter', 'worst area', 'worst concavity', 'worst fractal dimension']

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.

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 $\mathcal{S}(\mathcal{F}, k) = 2^k$.

We move down the values of k until we find a k where $\mathcal{S}(\mathcal{F}, k) \neq 2^k$.

- $\mathbf{k = 1}$, (x_1): Easily classifiable into $\{(+), (-)\}$, Thus $\mathcal{S} = 2$
- $\mathbf{k = 2}$, $(x_1 \leq x_2)$: Easily classifiable into $\{(++), (--)\}$
  - For $(+-)$ use $f(x) = \text{sgn}((x - (x_2 + 1))(x - \frac{x_1 + x_2}{2}))$
  - For $(-+)$ use $f(x) = \text{sgn}((x - (x_1 - 1))(x - \frac{x_1 + x_2}{2}))$

  Thus $\mathcal{S} = 4$
- $\mathbf{k = 3}$, $(x_1 \leq x_2 \leq x_3)$: Easily classifiable into $\{(+++), (---)\}$

- For $(+ + -)$ use $f(x) = \text{sgn}((x - (x_3 + 1))(x - \frac{x_2+x_3}{2}))$
- For $(+ - +)$ use $f(x) = \text{sgn}((x - \frac{x_1+x_2}{2})(x - \frac{x_2+x_3}{2}))$
- For $(- + +)$ use $f(x) = \text{sgn}((x - (x_1 - 1))(x - \frac{x_1+x_2}{2}))$
- For $(+ - -)$ use $f(x) = \text{sgn}((x - \frac{x_1+x_2}{2})(x - (x_3 + 1)))$
- For $(- - +)$ use $f(x) = \text{sgn}((x - (x_1 - 1))(x - \frac{x_2+x_3}{2}))$
- For $(- + -)$ use use $f(x) = \text{sgn}((\frac{x_1+x_2}{2} - x)(x - \frac{x_2+x_3}{2}))$

Thus $\mathcal{S} = 8$

- $\mathbf{k = 4}$, $(x_1 \leq x_2 \leq x_3 \leq x_4)$: Here we see that using the form $f(x) = \text{sgn}(ax^2 + bx + c)$ we can't classify $(+ - +-)$ in any way possible.

Therefore, $VC(\mathcal{F}) = 3$

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?

The above mentioned $\mathcal{F}$ is a classifier, classifying all points to the left of it as -1 and all to right of it as $+1$.

Lets arrange the points in a sequence $(x_{1'}, x_{2'}, ...x_{n'})$ where $x_{i'} \leq x_{j'} \ \forall \ i' < j'$. Using this we can show that:

$$\mathcal{S}(\mathcal{F}, n) = n + 1$$

This comes from the fact that the different ways to place a 1-d linear classifier is:

- $n - 1$ regions between the n sequentially arranged points
- 2 classifier on the extreme ends of the data-points i.e. $t < x_{1'}$ and $t > x_{n'}$

making a total $\mathbf{n + 1}$ 1-d classifier's.

The VC dimension:

$$VC(\mathcal{F}) = 1$$

for this definition of $\mathcal{F}$.

This can be calculated by equating $\mathcal{S}(\mathcal{F}, n) = n + 1 = 2^n$ and solving gives us $\mathbf{n = 1}$

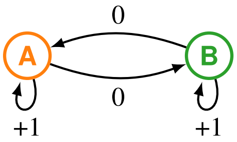
3. (6 pts) Show the following monotonicity property of VC-dimension: if $\mathcal{F}$ and $\mathcal{F}'$ are hypothesis classes with $\mathcal{F} \subset \mathcal{F}'$, then the VC-dimension of $\mathcal{F}'$ is greater than or equal to that of $\mathcal{F}$.

If $\mathcal{F}$ is a subset of $\mathcal{F}'$ then all the labelling sequences or shattering that are induced by class $\mathcal{F}$ are a part of $\mathcal{F}'$. Now considering $\mathcal{F}$ is a subset of $\mathcal{F}'$, $\mathcal{F}$ is some specific set of hypothesis class with tighter criteria to their form than $\mathcal{F}'$, hence $\mathcal{F}'$ with a looser form might be able to accommodate more labelling sequences or at-least have the same number of labelling sequences. Thus $\mathcal{F}'$ can have a higher VC-dimension or equivalent VC-dimension to $\mathcal{F}$ but never lower.

If we assume $\mathcal{F}'$ has a lower VC-dimension than $\mathcal{F}$ then there is some shattering in $\mathcal{F}$ which $\mathcal{F}'$ can't produce for a given n. This directly contradicts the fact $\mathcal{F} \subset \mathcal{F}'$ because if $\mathcal{F}$ can shatter a sequence then $\mathcal{F}'$ should be able to do it too.

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	3.972	4.999
B	3.998	4.977

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

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

$$s_t = \{A, B\}$$
$$a_t = \{\text{Stay}, \text{Move}\}$$

$s_1 = A$

Q_0	Move	Stay
A	0	0
B	0	0

Choosing 1^{th} Action $a_1 =$ Move: $s_2 = B$, $r_1 = 1$, $Q(A, \text{Move}) = 0.5$

Q_1	Move	Stay
A	0.5	0
B	0	0

Choosing 2^{th} Action $a_2 =$ Stay: $s_3 = B$, $r_2 = 0$, $Q(B, \text{Stay}) = 0.0$

Q_2	Move	Stay
A	0.5	0
B	0	0.0

Choosing 3^{th} Action $a_3 =$ Move: $s_4 = A$, $r_3 = 1$, $Q(B, \text{Move}) = 0.7$

Q_3	Move	Stay
A	0.5	0
B	0.7	0.0

Choosing 4^{th} Action $a_4 =$ Stay: $s_5 = A$, $r_4 = 0$, $Q(A, \text{Stay}) = 0.2$

Q_4	Move	Stay
A	0.5	0.2
B	0.7	0.0

Choosing 5^{th} Action $a_5 =$ Move: $s_6 = B$, $r_5 = 1$, $Q(A, \text{Move}) = 1.03$

Q_5	Move	Stay
A	1.03	0.2
B	0.7	0.0

Choosing 6^{th} Action $a_6 =$ Stay: $s_7 = B$, $r_6 = 0$, $Q(B, \text{Stay}) = 0.28$

Q_6	Move	Stay
A	1.03	0.2
B	0.7	0.28

Choosing 7^{th} Action $a_7 =$ Stay: $s_8 = B$, $r_7 = 0$, $Q(B, \text{Stay}) = 0.42$

Q_7	Move	Stay
A	1.03	0.2
B	0.7	0.42

Choosing 8^{th} Action $a_8 =$ Move: $s_9 = A$, $r_8 = 1$, $Q(B, \text{Move}) = 1.262$

Q_8	Move	Stay
A	1.03	0.2
B	1.262	0.42

Choosing 9^{th} Action $a_9 =$ Move: $s_{10} = B$, $r_9 = 1$, $Q(A, \text{Move}) = 1.52$

Q_9	Move	Stay
A	1.52	0.2
B	1.262	0.42

Choosing 10^{th} Action $a_{10} =$ Stay: $s_{11} = B$, $r_{10} = 0$, $Q(B, \text{Stay}) = 0.715$

Q_{10}	Move	Stay
A	1.52	0.2
B	1.262	0.715

Choosing 11^{th} Action $a_{11} =$ Stay: $s_{12} = B$, $r_{11} = 0$, $Q(B, \text{Stay}) = 0.862$

Q_{11}	Move	Stay
A	1.52	0.2
B	1.262	0.862

Choosing 12^{th} Action $a_{12} =$ Move: $s_{13} = A$, $r_{12} = 1$, $Q(B, \text{Move}) = 1.739$

Q_{12}	Move	Stay
A	1.52	0.2
B	1.739	0.862

Choosing 13^{th} Action $a_{13} =$ Stay: $s_{14} = A$, $r_{13} = 0$, $Q(A, \text{Stay}) = 0.708$

Q_{13}	Move	Stay
A	1.52	0.708
B	1.739	0.862

Choosing 14^{th} Action a_{14} = Move: $s_{15} = B$, $r_{14} = 1$, $Q(A, \text{Move}) = 1.955$

Q_{14}	Move	Stay
A	1.955	0.708
B	1.739	0.862

Choosing 15^{th} Action a_{15} = Stay: $s_{16} = B$, $r_{15} = 0$, $Q(B, \text{Stay}) = 1.127$

Q_{15}	Move	Stay
A	1.955	0.708
B	1.739	1.127

Choosing 16^{th} Action a_{16} = Stay: $s_{17} = B$, $r_{16} = 0$, $Q(B, \text{Stay}) = 1.259$

Q_{16}	Move	Stay
A	1.955	0.708
B	1.739	1.259

Choosing 17^{th} Action a_{17} = Stay: $s_{18} = B$, $r_{17} = 0$, $Q(B, \text{Stay}) = 1.325$

Q_{17}	Move	Stay
A	1.955	0.708
B	1.739	1.325

Choosing 18^{th} Action a_{18} = Move: $s_{19} = A$, $r_{18} = 1$, $Q(B, \text{Move}) = 2.152$

Q_{18}	Move	Stay
A	1.955	0.708
B	2.152	1.325

Choosing 19^{th} Action a_{19} = Stay: $s_{20} = A$, $r_{19} = 0$, $Q(A, \text{Stay}) = 1.136$

Q_{19}	Move	Stay
A	1.955	1.136
B	2.152	1.325

Choosing 20^{th} Action a_{20} = Stay: $s_{21} = A$, $r_{20} = 0$, $Q(A, \text{Stay}) = 1.35$

Q_{20}	Move	Stay
A	1.955	1.35
B	2.152	1.325

Choosing 21^{th} Action a_{21} = Stay: $s_{22} = A$, $r_{21} = 0$, $Q(A, \text{Stay}) = 1.457$

Q_{21}	Move	Stay
A	1.955	1.457
B	2.152	1.325

Choosing 22^{th} Action a_{22} = Move: $s_{23} = B$, $r_{22} = 1$, $Q(A, \text{Move}) = 2.338$

Q_{22}	Move	Stay
A	2.338	1.457
B	2.152	1.325

Choosing 23th Action $a_{23} =$ Move: $s_{24} = A$, $r_{23} = 1$, $Q(B, \text{Move}) = 2.511$

Q_{23}	Move	Stay
A	2.338	1.457
B	2.511	1.325

Choosing 24th Action $a_{24} =$ Move: $s_{25} = B$, $r_{24} = 1$, $Q(A, \text{Move}) = 2.674$

Q_{24}	Move	Stay
A	2.674	1.457
B	2.511	1.325

Choosing 25th Action $a_{25} =$ Stay: $s_{26} = B$, $r_{25} = 0$, $Q(B, \text{Stay}) = 1.667$

Q_{25}	Move	Stay
A	2.674	1.457
B	2.511	1.667

Choosing 26th Action $a_{26} =$ Move: $s_{27} = A$, $r_{26} = 1$, $Q(B, \text{Move}) = 2.825$

Q_{26}	Move	Stay
A	2.674	1.457
B	2.825	1.667

Choosing 27th Action $a_{27} =$ Stay: $s_{28} = A$, $r_{27} = 0$, $Q(A, \text{Stay}) = 1.798$

Q_{27}	Move	Stay
A	2.674	1.798
B	2.825	1.667

Choosing 28th Action $a_{28} =$ Move: $s_{29} = B$, $r_{28} = 1$, $Q(A, \text{Move}) = 2.967$

Q_{28}	Move	Stay
A	2.967	1.798
B	2.825	1.667

Choosing 29th Action $a_{29} =$ Move: $s_{30} = A$, $r_{29} = 1$, $Q(B, \text{Move}) = 3.099$

Q_{29}	Move	Stay
A	2.967	1.798
B	3.099	1.667

Choosing 30th Action $a_{30} =$ Stay: $s_{31} = A$, $r_{30} = 0$, $Q(A, \text{Stay}) = 2.086$

Q_{30}	Move	Stay
A	2.967	2.086
B	3.099	1.667