Practical Lecture 6 - VC Dimension, Support Vector Machines

Luis Sa-Couto[1] and Andreas Wichert[2]

INESC-ID, Instituto Superior Tecnico, Universidade de Lisboa
`{luis.sa.couto,andreas.wichert}@tecnico.ulisboa.pt`

1 VC Dimension

The VC dimension is a way to measure the degrees of freedom of a classifier family. To make clear how to compute it, we will use as an example the set of 1 dimensional interval binary classifiers:

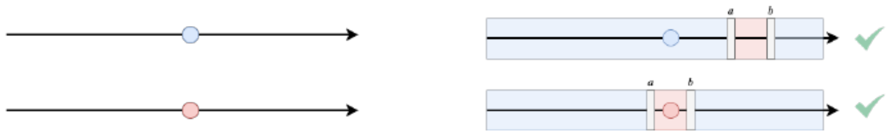
$$h(x; a, b) = \begin{cases} 1 & a \geq x \geq b \\ 0 & \textit{otherwise} \end{cases}$$

Before going any further we need to recall the concept of a classifier family shattering n points.

Think about a game between you and an adversarial nature. In this game, you "play" first and take n points and place them anywhere you like. Afterwards, nature places targets (binary $+1$ or -1) on each point, this choice is called a dichotomy. Finally, you have the last move. If you can choose a classifier from the family in question that is able to correctly classify every point you win.

Having understood the rules of the game, the concept of shattering is simple. A classifier family can shatter n points if you can win at this game. More specifically, if you can find n points such that, regardless of the way nature chooses the targets, you can always choose a classifier from the family that solves the problem.

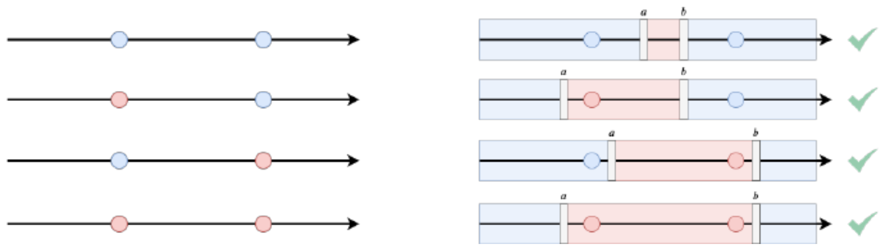
Let us look at our example. We would like to check if our family of classifiers can shatter one point. In the figure below, we can see that we are able to choose a point that is correctly classified for all possible targets that nature can give it.



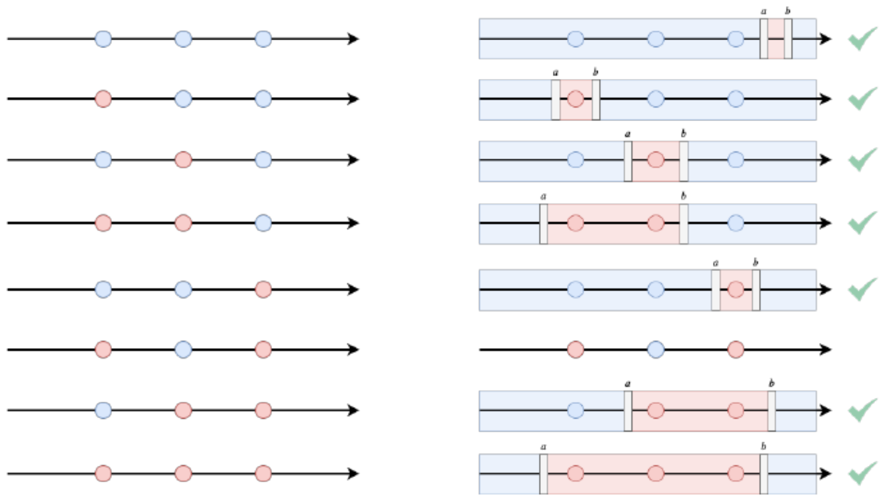
But what is the relation between shattering and the VC dimension? Well, if a classifier family h shatters n points then we say that $d_{VC}(h) \geq n$. So, from our example follows that $d_{VC}(h) \geq 1$.

However, this game only allows us to compute lower bounds for the VC dimension. So, how can we get a final value for it?

We start by finding the tightest lower bound possible. In our example, this means increasing n until it is not obvious we can win. Let do it for $n = 2$:



We see that we can win regardless of the targets, so $d_{VC}(h) \geq 2$. However, by increasing n again we see that it is not so easy:



So, we need to prove that it is not possible to win the game when $n = 3$.

Given any three points $x_1 < x_2 < x_3$, the problematic dichotomy is such that $t_1 = +1, t_2 = -1, t_3 = +1$ since all others can be easily solved like in the figure above. If we try to place the interval, we either:

- Classify correctly x_1 and x_2 but miss x_3 using an interval around x_1: $a < x_1 < b < x_2 < x_3$
- Classify correctly x_3 and x_2 but miss x_1 using an interval around x_3: $x_1 < x_2 < a < x_3 < b$
- Classify correctly x_1 and x_3 but miss x_2 using an interval around the three points: $a < x_1 < x_2 < x_3 < b$

So, there is no way to correctly classify this dichotomy which means we cannot shatter three points and so we say that $d_{VC}(h) < 3$. Putting our lower bound together with the upper bound we conclude that $d_{VC}(h) = 2$.

With the example it becomes simple to see that the VC dimension of a classifier family equals the maximum number of points that it can shatter.

A simple way to get an approximation for the VC dimension is to count the number of free parameters in a classifier family. This number will sometimes not yield the correct value for the VC dimension since it can be the case that not all parameters correspond to degrees of freedom. However, it is generally a good heuristic to think about.

From knowing the VC dimension, several key insights can follow. On the one hand, a large VC dimension will mean a lot of capacity to accomodate to small changes in the data which can lead to overfitting. This relates to a high variance in the bias-variance view. In such scenario we will need more data to get a good parameter estimation and most likely regularization will be useful. On the other hand, a low VC dimension will restrict the freedom of the classifier to adjust to the data imposing a bias. However, the restricted freedom allows us to work with less data and possibly even do not require regularization. In reality, what high and low VC dimension means highly depends on the problem itself. In fact, it is only through exeprimental practice with a given classifier family that one can judge if its VC dimension is too low, too big or just right for the problem at hand.

1) Show graphically what is the VC dimension of 1 dimensional threshold binary classifiers:

$$h(x;a) = \begin{cases} 1 & x \geq a \\ 0 & x < a \end{cases}$$

Solution:

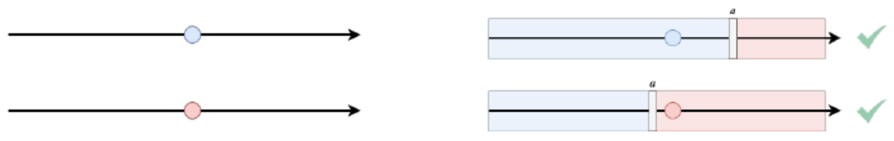
To find the VC dimension of a classifier we need two key steps:

1) To prove that $d_{VC} \geq N$ we need to find a set of N points that can be shatter by the classifier.

2) To prove that $d_{VC} \leq N$ we need to show that it is not possible to find a set of $N+1$ points that can be shatter by the classifier.

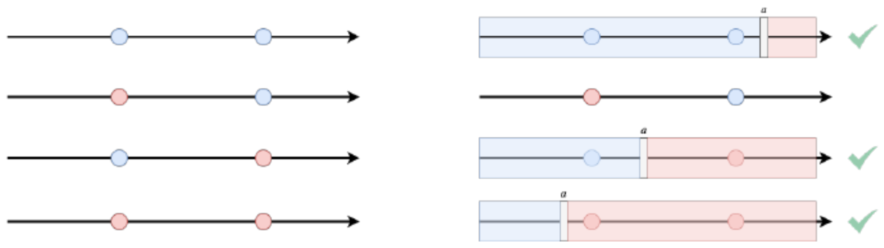
Having stated that, we want to find a lower bound on the VC dimension. To do that, we apply step 1 for $N = 1$ and then repeat it for increasing values of N.

So, for $N = 1$ we can find an example that can be shattered by h:



Thus, we can conclude $d_{VC} \geq 1$.

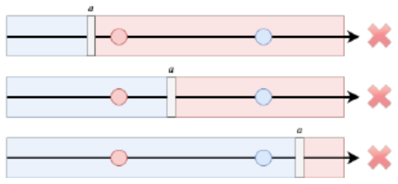
Now, for $N = 2$ it is not so easy. Specifically, one dichotomy is problematic.



We cannot easily complete step 1. So, we try to get an upper-bound from step 2. Given any two points, there are three places where we can place a:

1. On the left of the two points
2. In the middle of the two points
3. On the right of the two points

Now, for the dichotomy where the target of the leftmost point is $+1$ and the other is -1 it is not possible to correctly separate the two classes:



With that, we conclude that $d_{VC} \leq 1$.

Putting all together, we get $d_{VC} = 1$.

---

2) Show graphically what is the VC dimension of 2 dimensional axis-aligned rectangle binary classifiers:

$$h\left(x_1, x_2; a_1, a_2, h, w\right) = \begin{cases} 1 & a_1 + w \geq x_1 \geq a_1, a_2 + h \geq x_2 \geq a_2 \\ 0 & otherwise \end{cases}$$

Solution:

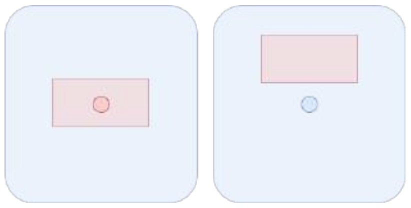
To find the VC dimension of a classifier we need two key steps:

1) To prove that $d_{VC} \geq N$ we need to find a set of N points that can be shatter by the classifier.

2) To prove that $d_{VC} \leq N$ we need to show that it is not possible to find a set of $N+1$ points that can be shatter by the classifier.

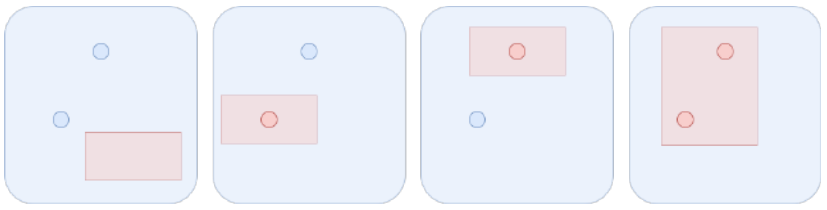
Having stated that, we want to find a lower bound on the VC dimension. To do that, we apply step 1 for $N = 1$ and then repeat it for increasing values of N.

So, for $N = 1$ we can find an example that can be shattered by h:



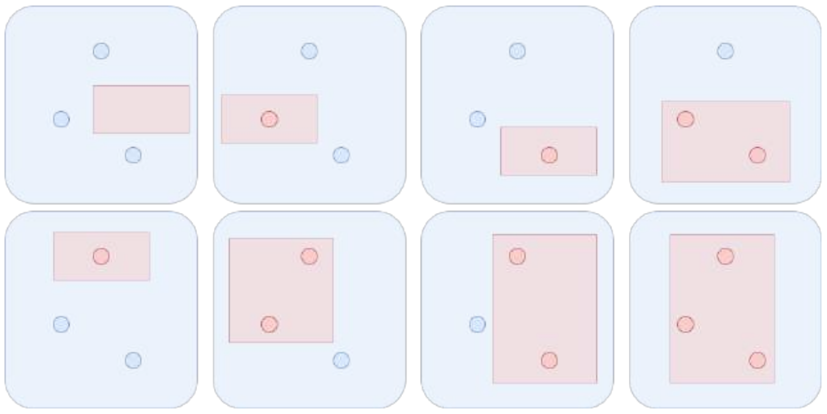
Thus, we can conclude $d_{VC} \geq 1$.

For $N = 2$ we can find an example that can be shattered by h:



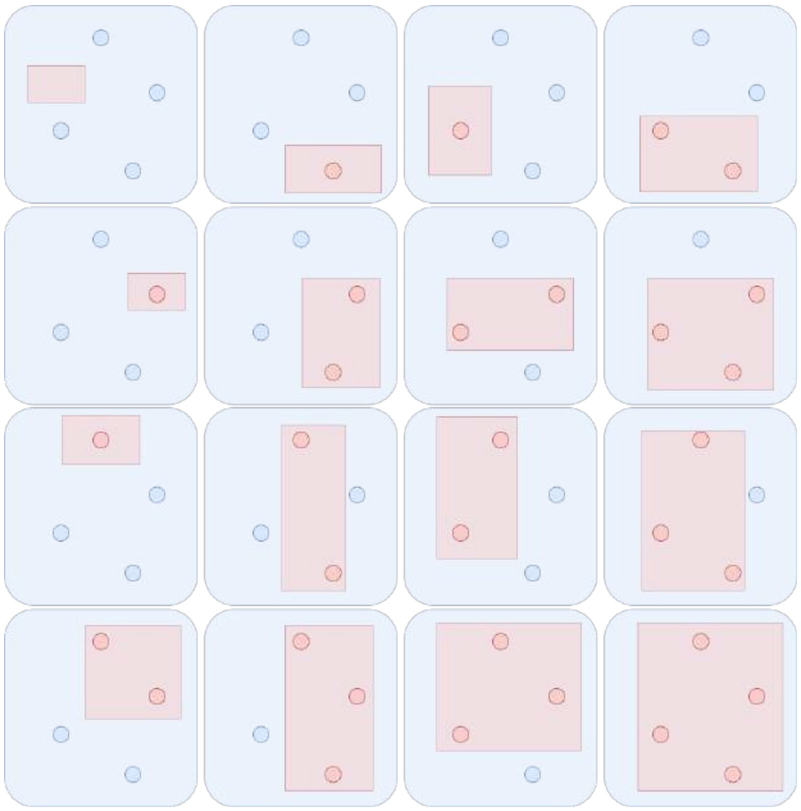
Thus, we can conclude $d_{VC} \geq 2$.

For $N = 3$ we can find an example that can be shattered by h:



Thus, we can conclude $d_{VC} \geq 3$.

For $N = 4$ we can find an example that can be shattered by h:



Thus, we can conclude $d_{VC} \geq 4$.

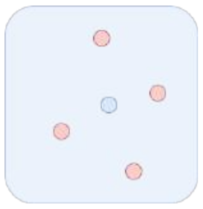
Now, for $N = 5$ it is not so easy. We cannot easily complete step 1. So, we try to get an upper-bound from step 2.

For any five points there are two possibilities:

1. There are two or more points forming a horizontal or vertical line
2. Any rectangle having four of the points in its edges necessarily encloses the remaining point its interior

We cannot shatter scenario 1 for the same reasons presented in the previous exercise.

For scenario 2, consider the dichotomy where all four points that form the quadrilateral have $t_i = +1$ and the point in the middle has $t = 1$ like in the figure below.



There is no way to classify all quadrilateral points correctly without missclassifying the middle point.

So, there is no way to correctly classify this dichotomy. Which means we cannot shatter five points. With that, we conclude that $d_{VC} \leq 4$.

Putting all together, we get $d_{VC} = 4$.

---

3) Show graphically what is the VC dimension of 2 dimensional perceptrons:

$$h(x_1, x_2; w_0, w_1, w_2) = sgn(w_0 + w_1 x_1 + w_2 x_2)$$

Solution:

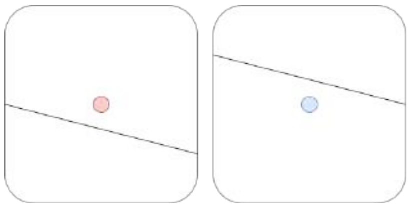
To find the VC dimension of a classifier we need two key steps:

1) To prove that $d_{VC} \geq N$ we need to find a set of N points that can be shatter by the classifier.

2) To prove that $d_{VC} \leq N$ we need to show that it is not possible to find a set of $N+1$ points that can be shatter by the classifier.

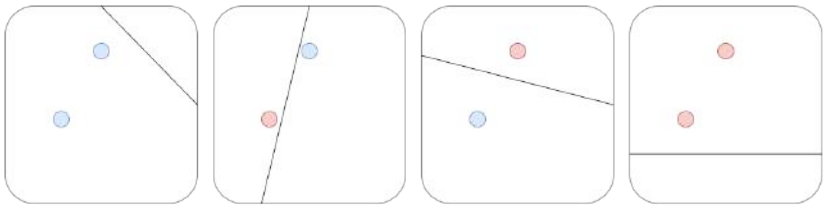
Having stated that, we want to find a lower bound on the VC dimension. To do that, we apply step 1 for $N = 1$ and then repeat it for increasing values of N.

So, for $N = 1$ we can find an example that can be shattered by h:



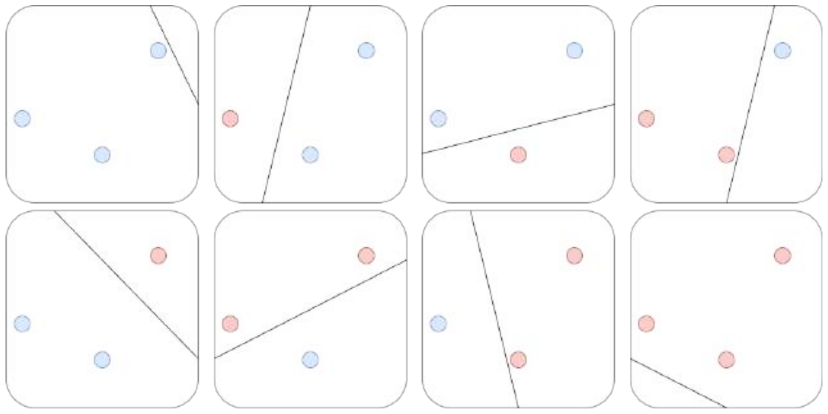
Thus, we can conclude $d_{VC} \geq 1$.

For $N = 2$ we can find an example that can be shattered by h:



Thus, we can conclude $d_{VC} \geq 2$.

For $N = 3$ we can find an example that can be shattered by h:



Thus, we can conclude $d_{VC} \geq 3$.

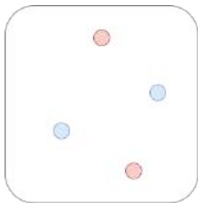
Now, for $N = 4$ it is not so easy. We cannot easily complete step 1. So, we try to get an upper-bound from step 2.

For any four points there are two possibilities:

1. There are collinear points
2. The four points form a quadrilateral

We cannot shatter scenario 1 for the same reasons presented in the first exercise.

For scenario 2, consider a dichotomy that resembles the XOR problem like in the figure below.



There is no way to classify all points correctly. Which means we cannot shatter four points. With that, we conclude that $d_{VC} \leq 3$.

Putting all together, we get $d_{VC} = 3$.

---

4) Show analytically that the VC dimension of a d dimensional perceptron:

$$h(\mathbf{x}; \mathbf{w}) = sgn\left(\mathbf{w}^T \mathbf{x}\right)$$

where $\mathbf{x} = \left(1\ x_1\ \ldots\ x_d\right)^T$ and $\mathbf{w} = \left(w_0\ w_1\ \ldots\ w_d\right)^T$ is $d_{VC} = d + 1$.

Solution:

Step 1: Prove that there exists a set of $d+1$ points that the perceptron can shatter.

For a dataset with the data points $\mathbf{x}_i = \begin{pmatrix} x_1 \; x_2 \; \dots \; x_d \end{pmatrix}^T$ placed on the rows (adding a bias dimension) of a matrix:

$$\mathbf{X} = \begin{pmatrix} 1 & -\mathbf{x}_1^T- \\ 1 & -\mathbf{x}_2^T- \\ 1 & -\mathbf{x}_3^T- \\ 1 & \vdots \\ 1 & -\mathbf{x}_{d+1}^T- \end{pmatrix}$$

With targets $t_i \in \{-1, 1\}$ placed on a column vector:

$$\mathbf{t} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_{d+1} \end{pmatrix}$$

We need to prove that the perceptron can shatter this problem. To achieve that, all we need to do is to find a weight vector $\mathbf{w}^*$ such that the perceptron correctly classifies all possible targets (all dichotomies). This means that:

$$sgn\left(\mathbf{X}\mathbf{w}^*\right) = \mathbf{t}$$

A possible solution is $\mathbf{w}^* = \mathbf{X}^{-1}\mathbf{t}$, because:

$$sgn\left(\mathbf{X}\mathbf{w}^*\right) = sgn\left(\mathbf{X}\mathbf{X}^{-1}\mathbf{t}\right) = sgn\left(\mathbf{I}\mathbf{t}\right) = sgn\left(\mathbf{t}\right) = \mathbf{t}$$

Having showed this, all we need to do is to find a data matrix $\mathbf{X}$ that is invertible. An example that works is:

$$X = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

One check this by noticing that the columns form a base for R^{d+1} space. Which means that no column can be written as linear combination of the others.

So, since we have one example that works we conclude our proof and thus $d_{VC} \geq d+1$.

Step 2: Prove that perceptron cannot shatter ANY $d+2$ points.

Consider any $d+2$ points: $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_d, \mathbf{x}_{d+1}, \mathbf{x}_{d+2}$.

We have more points than dimensions, so we must have:

$$\mathbf{x}_j = \sum_{i \neq j} a_i \mathbf{x}_i$$

where at least some $a_i \neq 0$.

Which can be rewritten as follows to make the net inputs appear:

$$\mathbf{x}_j = \sum_{i \neq j} a_i \mathbf{x}_i$$

$$\mathbf{w}^T \mathbf{x}_j = \mathbf{w}^T \sum_{i \neq j} a_i \mathbf{x}_i$$

$$\mathbf{w}^T \mathbf{x}_j = \sum_{i \neq j} a_i \mathbf{w}^T \mathbf{x}_i$$

To prove that the perceptron cannot shatter any $d+2$ points we need to find a dichotomy it cannot learn. So, let us consider the following dichotomy:

1. All $\mathbf{x}_i$ with $a_i \neq 0$ get $t_i = sgn\,(a_i)$
2. $\mathbf{x}_j$ gets $t_j = -1$

For the perceptron to be correct, we need that its output equals the target:

$$t_i = sgn\,(a_i) = sgn\left(\mathbf{w}^T \mathbf{x}_i\right)$$

So, if a_i and $\mathbf{w}^T \mathbf{x}_i$ have the same sign, the product between the two will be positive:

$$a_i \mathbf{w}^T \mathbf{x}_i > 0$$

Which means that the sum for all points except j is also positive:

$$\sum_{i \neq j} a_i \mathbf{w}^T \mathbf{x}_i > 0$$

Which means that:

$$\mathbf{w}^T \mathbf{x}_j = \sum_{i \neq j} a_i \mathbf{w}^T \mathbf{x}_i > 0$$

Thus, the target of j needs to be positive:

$$t_j = sgn\left(\mathbf{w}^T \mathbf{x}_j\right) = +1$$

Which is an error! So, no perceptron can classify this dichotomy. Which means that the perceptron cannot shatter any set of $d+2$ points. So, $d_{VC} \leq d+1$.

Step 3: Combining the two steps

Combining both steps, we get:

$$d_{VC} \geq d + 1 \wedge d_{VC} \leq d + 1$$

$$d + 1 \leq d_{VC} \leq d + 1 \Rightarrow d_{VC} = d + 1$$

---

5) Show analytically that the VC dimension of a Decision tree on inputs with d boolean features is 2^d.

Solution:

The first thing we need to do is to notice that 2^d is the maximum number of different points that can exist in this scenario. For that reason, we immediately know that $d_{VC} \leq 2^d$ for any classifier with this kind of input.

Noticing that, all we need to do is to create a decision tree that can shatter a data set with all possible points.

Using simple binary splits on all features, we can create a decision tree with height $d+1$ that has one leaf for each point. Each leaf will have the label of that point. So, all points will be classified correctly regardless of the dichotomy chosen.

This proves that $d_{VC} \geq 2^d$.

Again, we can combine both facts and get:

$$d_{VC} \geq 2^d \wedge d_{VC} \leq 2^d$$

$$2^d \leq d_{VC} \leq 2^d \Rightarrow d_{VC} = 2^d$$

---

6) For the following scenarios which would you **approximately** say has smallest VC dimension?

a) Three dimensional real inputs classified by:

1. MLP with one hidden layer with the following units per layer $\begin{pmatrix}3 & 2 & 2\end{pmatrix}$.
2. Simple Bayesian classifier with multivariate gaussian likelihood function.

Solution:

We know that the VC dimension is a measure of the degrees of freedom of a classifier. A good proxy for the number of degrees of freedom is the number of parameters. So, let us estimate the number of parameters for each scenarion.

1.

Between the input layer and the hidden layer we will have:

- A 2×3 weight matrix $\mathbf{W}^{[1]}$, so 6 parameters
- A 2×1 bias vector $\mathbf{b}^{[1]}$, so 2 parameters

Between the hidden layer and the output layer we will have:

- A 2×2 weight matrix $\mathbf{W}^{[2]}$, so 4 parameters
- A 2×1 bias vector $\mathbf{b}^{[2]}$, so 2 parameters

There are no more parameters, so the total amounts to $6 + 2 + 4 + 2 = 14$.

2.

For the simple Bayesian classifier, we need to estimate prior and likelihood.

The prior is a distribution table with two entries (one for each class):

$P(t=0)$	$P(t=1)$
p	$1-p$

So, we have one parameter for the prior.

The likelihood $p(\mathbf{x} \mid t=0)$, being a multivariate gaussian, will require a mean vector and a covariance matrix:

- For three dimensions the mean vector is a 3×1 vector, so 3 parameters.
- For three dimensions, the covariance is a 3×3 matrix $\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} & \Sigma_{13} \\ \Sigma_{21} & \Sigma_{22} & \Sigma_{23} \\ \Sigma_{31} & \Sigma_{32} & \Sigma_{33} \end{pmatrix}$, however the matrix is symmetric so, we only need to count the diagonal and upper diagonal part of the matrix. So, 6 parameters.

The likelihood $p(\mathbf{x} \mid t=1)$, being a multivariate gaussian, requires the same number of parameters.

There are no more parameters to estimate, so the total is $2\times(3+6)+1 = 19$.

Finally, we can produce our guess on the VC dimensions of both classifiers. We would approximately say that the MLP classifier has a smaller VC dimension $d_{VC}(MLP) < d_{VC}(Bayesian)$.

---

b) Four dimensional boolean inputs classified by:

1. Decision Tree.
2. Naive Bayes.

Solution:

We know that the VC dimension is a measure of the degrees of freedom of a classifier. A good proxy for the number of degrees of freedom is the number of parameters. So, let us estimate the number of parameters for each scenario.

1.

We know from exercise 6 that $d_{VC}(DecisionTree) = 2^d$. So, in this case, we have that $d_{VC}(DecisionTree) = 2^4 = 16$

2.

For the Naive Bayesian classifier, we need to estimate prior and likelihoods.

The prior is a distribution table with two entries (one for each class):

$P(t=0)$	$P(t=1)$
p	$1-p$

So, we have one parameter for the prior.

The likelihoods, like the prior are distributions over binary variables so for each class c and for each feature i we have a distribution $p(x_i \mid t = c)$ that requires one parameter.

Since there are 2 classes and 4features, we have $2 \times 4 \times 1 = 8$ parameters for the likelihoods.

There are no more parameters to estimate, so the total is $1 + 8 = 9$.

Finally, we can produce our guess on the VC dimensions of both classifiers. We would approximately say that the Naive Bayes classifier has a smaller VC dimension $d_{VC}(NB) < d_{VC}(DT)$.

---

c) N-dimensional real inputs classified by:

1. Naive Bayes with Gaussian likelihoods.
2. MLP with two hidden layers with the following units per layer $\left(N \; \frac{N}{2} \; \frac{N}{2} \; 2\right)$.
3. Simple Bayesian classifier with multivariate gaussian likelihood function.
4. Perceptron.

Solution:

We know that the VC dimension is a measure of the degrees of freedom of a classifier. A good proxy for the number of degrees of freedom is the number of parameters. So, let us estimate the number of parameters for each scenario.

1.

For the Naive Bayesian classifier, we need to estimate prior and likelihoods.

The prior is a distribution table with two entries (one for each class):

$P(t = 0)$	$P(t = 1)$
p	$1 - p$

So, we have one parameter for the prior.

The likelihoods are one dimensional gaussians. Each one dimensional gaussian requires two parameters: mean and standard deviation. So for each class c and for each feature i we have a distribution $p(x_i \mid t = c)$ that requires two parameters.

Since there are 2 classes and N features, we have $2 \times N \times 2 = 4N$ parameters for the likelihoods.

There are no more parameters to estimate, so the total is $1 + 4N$.

2.

Between the input layer and the first hidden layer we will have:

– A $\frac{N}{2} \times N$ weight matrix $\mathbf{W}^{[1]}$, so $\frac{N^2}{2}$ parameters
– A $\frac{N}{2} \times 1$ bias vector $\mathbf{b}^{[1]}$, so $\frac{N}{2}$ parameters

Between the first hidden layer and the second one we will have:

– A $\frac{N}{2} \times \frac{N}{2}$ weight matrix $\mathbf{W}^{[2]}$, so $\frac{N^2}{4}$ parameters
– A $\frac{N}{2} \times 1$ bias vector $\mathbf{b}^{[2]}$, so $\frac{N}{2}$ parameters

Between the second hidden layer and the output layer we will have:

- A $2 \times \frac{N}{2}$ weight matrix $\mathbf{W}^{[3]}$, so N parameters
- A 2×1 bias vector $\mathbf{b}^{[3]}$, so 2 parameters

There are no more parameters, so the total amounts to $\frac{N^2}{2}+\frac{N}{2}+\frac{N^2}{4}+\frac{N}{2}+N+2 = \frac{2N^2+2N+N^2+2N+4N+8}{4} = \frac{3N^2+8N+8}{4}$.

3.

For the simple Bayesian classifier, we need to estimate prior and likelihood. The prior is a distribution table with two entries (one for each class):

$P(t=0)$	$P(t=1)$
p	$1-p$

So, we have one parameter for the prior.

The likelihood $p(\mathbf{x} \mid t=0)$, being a multivariate gaussian, will require a mean vector and a covariance matrix:

- For N dimensions the mean vector is an $N \times 1$ vector, so N parameters.
- For three dimensions, the covariance is an $N \times N$ matrix, however the matrix is symmetric so, we only need to count the diagonal and upper diagonal part of the matrix. So, $N + \frac{N^2-N}{2}$ parameters.

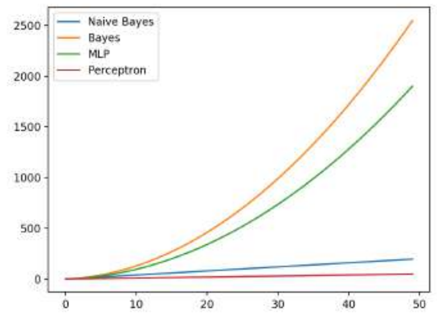
The likelihood $p(\mathbf{x} \mid t=1)$, being a multivariate gaussian, requires the same number of parameters.

There are no more parameters to estimate, so the total is $1+2\times\left(N+N+\frac{N^2-N}{2}\right) = 1+\left(2N+2N+N^2-N\right) = 1+3N+N^2$.

4.

The perceptron has a weight vector $\mathbf{w} = \begin{pmatrix} w_1 & w_2 & \cdots & w_N \end{pmatrix}^T$ and a bias w_0. So, it has $N+1$ parameters.

Finally, we can plot our guesses on the VC dimensions of the four scenarios:

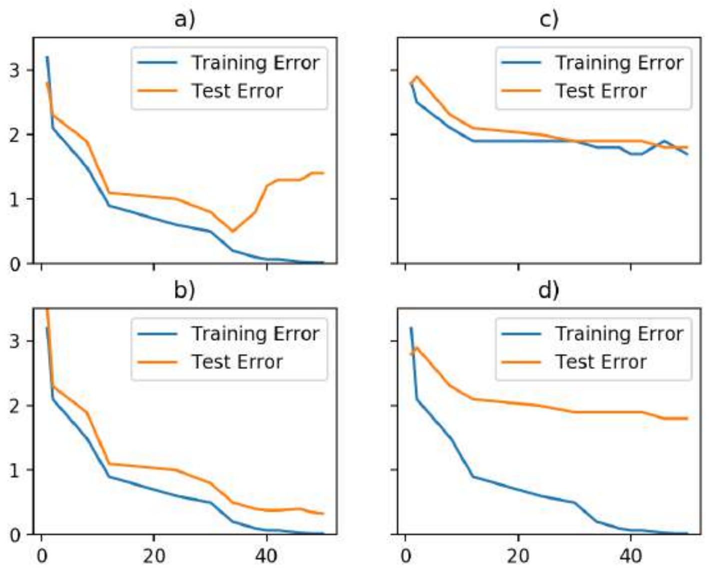


We would approximately order the VC dimensions as follows $d_{VC}(Perceptron) < d_{VC}(NB) < d_{VC}(MLP) < d_{VC}(Bayes)$.

7) a) Choose between **increase**, **decrease**, **maintain** for each of the following factors:

– Training data
– Regularization
– VC dimension

For each of the following four scenarios:



Justify every decision.

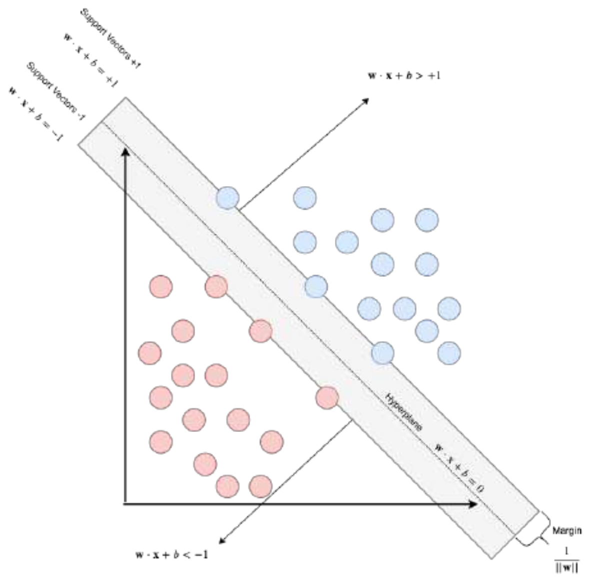
Solution:
a) Increase, increase, maintain/decrease
b) Increase, maintain, maintain
c) Increase, decrease, increase
d) Increase, increase, maintain/decrease

---

2 Support Vector Machines

The original support vector machine (SVM) is a binary classifier that finds an hyperplane that separates two classes. However, unlike perceptrons and logistic regressions it does not find any hyperplane that separates the two classes. SVMs look for the best possible hyperplane. But what does this mean?

To explain the process we will use the following figure with a two-dimensional example:



Between two sets of points from two classes, only a few will be on close to the frontier. These points are called the support vectors. The separation hyperplane defined by the weight vector **w** and bias term b will have to be placed somewhere between these.

The key idea is to maximize the distance between the boundary and both classes. To measure this distance, we have a quantity called the margin m which is defined as the distance between the support vectors and the boundary and can be computed through $m = \frac{1}{\|\mathbf{w}\|_2}$.

So, the learning procedure would be to find the weight vector that maximizes m. However, we need to restrict the optimization procedure such that the boundary hyperplane correctly classifies the points. To that end, we define the constraint:

$$\begin{cases} \mathbf{w}^T\mathbf{x}^{(i)} + b = +1 & t^{(i)} = +1 \\ \mathbf{w}^T\mathbf{x}^{(i)} + b = -1 & t^{(i)} = -1 \end{cases} \Longleftrightarrow t^{(i)} \left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right) \geq 1$$

where $\mathbf{x}^{(i)}$ is a support vector and $t^{(i)}$ is its target.

There are two ways to solve the optimization problem. The primal way finds **w** and b directly. Using it, we get the parameters directly and, so, after learning, the SVM classifies points with:

$$SVM\left(\mathbf{x}; \mathbf{w}, b\right) = sgn\left(\mathbf{w}^T\mathbf{x} + b\right)$$

The dual way is sometimes easier to optimize and uses the support vectors directly. Each point $\mathbf{x}^{(i)}$ gets a coefficient α_i which determines its contribution to the boundary. So, all non support vectors will have a zero coefficient. With this formulation, the optimizer tries to find the coefficients and, in the end, the SVM uses them to classify points as follows:

$$SVM\left(\mathbf{x};\mathbf{w},b\right) = sgn\left(\sum_{i=1}^{N} t^{(i)}\alpha_i \mathbf{x}^T\mathbf{x}^{(i)} + b\right)$$

As we can see, this dual version writes the decision function for a point based on its inner product with the training vectors. It is in this formulation that we can see where the kernel trick enters. If we replace the dot product $\mathbf{x}^T\mathbf{x}^{(i)}$ by a kernel function $K\left(\mathbf{x},\mathbf{x}^{(i)}\right)$ we can interpret the kernel as an inner product in a high dimensional space where the problem is separable.

1) Consider the training set:

$$\mathbf{x}^{(1)} = \begin{pmatrix}0\\0\\2\end{pmatrix}, \mathbf{x}^{(2)} = \begin{pmatrix}0\\1\\8\end{pmatrix}, \mathbf{x}^{(3)} = \begin{pmatrix}1\\0\\6\end{pmatrix}, \mathbf{x}^{(4)} = \begin{pmatrix}1\\1\\7\end{pmatrix}, \mathbf{x}^{(5)} = \begin{pmatrix}1\\1\\3\end{pmatrix}$$

With targets:

$$t^{(1)} = +1, t^{(2)} = +1, t^{(3)} = -1, t^{(4)} = -1, t^{(5)} = +1$$

After training an SVM, we get the following coefficients:

$$\alpha^{(1)} = 0, \alpha^{(2)} = 1, \alpha^{(3)} = 0.5, \alpha^{(4)} = 1, \alpha^{(5)} = 0.5$$

a) How many support vectors are required?

Solution:
Four. All that have $\alpha_i > 0$.

---

b) Using a linear kernel and $b = -3$, how does this SVM classify $\mathbf{x} = \left(1\,1\,8\right)^T$.

Solution:

$$\begin{aligned}
o(\mathbf{x}) &= sgn\left(\sum_{i=1}^{4} t^{(i)}\alpha_i \mathbf{x}^T\mathbf{x}^{(i)} + b\right)\\
&= sgn((+1)(1)\begin{pmatrix}1 & 1 & 8\end{pmatrix}\begin{pmatrix}0\\1\\8\end{pmatrix} + (-1)\left(\frac{1}{2}\right)\begin{pmatrix}1 & 1 & 8\end{pmatrix}\begin{pmatrix}1\\0\\6\end{pmatrix}\\
&+(-1)(1)\begin{pmatrix}1 & 1 & 8\end{pmatrix}\begin{pmatrix}1\\1\\7\end{pmatrix} + (+1)\left(\frac{1}{2}\right)\begin{pmatrix}1 & 1 & 8\end{pmatrix}\begin{pmatrix}1\\1\\3\end{pmatrix} - 3)\\
&= sgn(65 - 24.5 - 58 + 13 - 3)\\
&= sgn(-7.5)\\
&= -1
\end{aligned}$$

---

2) Consider the training set put on a design matrix with the examples on the columns:

$$\mathbf{X} = \begin{pmatrix} 3.29 & 2.29 & 2.07 & 1.84 & 3.06 & 1.41 & 0.70 & 0.22 & 1.18 & 0.83\\ 1.60 & 1.81 & 0.91 & 1.37 & 1.66 & 0.51 & 1.25 & 0.35 & 0.25 & 0.62 \end{pmatrix}$$

With targets placed on a row vector:

$$\mathbf{t} = \begin{pmatrix} +1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

After training an SVM, we get the following coefficients:

$$\alpha = \begin{pmatrix} 0 & 0 & 3.36 & 0 & 0 & 3.36 & 0 & 0 & 0 & 0 \end{pmatrix}$$

a) How many support vectors are required?

Solution:
Two. All that have $\alpha_i > 0$.

---

b) What is the weight vector?

Solution:
Looking at the classification function:

$$o(\mathbf{x}) = sgn\left(\sum_{i=1}^{2} t^{(i)}\alpha_i \mathbf{x}^T\mathbf{x}^{(i)} + b\right)$$

We can rewrite is as:

$$o(\mathbf{x}) = sgn\left(\sum_{i=1}^{2} t^{(i)}\alpha_i \left(\mathbf{x}^{(i)}\right)^T \mathbf{x} + b\right)$$

Which can be written in terms of the weight vector as:

$$o(\mathbf{x}) = sgn\left(\mathbf{w}^T\mathbf{x} + b\right)$$

With:

$$\mathbf{w} = \sum_{i=1}^{2} t^{(i)}\alpha_i \mathbf{x}^{(i)}$$

So, we can get the weight vector:

$$\mathbf{w} = (+1)(3.36)\begin{pmatrix}2.07\\0.91\end{pmatrix} + (-1)(3.36)\begin{pmatrix}1.41\\0.51\end{pmatrix} = \begin{pmatrix}2.2176\\1.344\end{pmatrix}$$

---

c) What is the boundary equation?

Solution:

To get the boundary equation, we need the weight vector $\mathbf{w}$ and the bias b. We have the weight vector from the previous question. All we need is the bias.

We know that, for any support vector $\mathbf{x}^{(i)}$ with target $t^{(i)}$, we have that:

$$\mathbf{w}^T\mathbf{x}^{(i)} + b = t^{(i)}$$

So, we can take a support vector like $\mathbf{x}^{(3)} = \begin{pmatrix}2.07\\0.91\end{pmatrix}$ and use it to get the bias:

$$\begin{aligned}\mathbf{w}^T\mathbf{x}^{(i)} + b &= t^{(i)}\\ \begin{pmatrix}2.2176\\1.344\end{pmatrix}^T \begin{pmatrix}2.07\\0.91\end{pmatrix} + b &= +1\\ b &= 1 - 5.81\\ b &= -4.81\end{aligned}$$

Now we have all we need:

$$\mathbf{w}^T\mathbf{x} + b = 0 \Longleftrightarrow 2.2176x_1 + 1.344x_2 - 4.81 = 0$$

---

d) What is the value of the margin?

Solution:

For the boundary line we have:

$$\mathbf{w}^T\mathbf{x} + b = 0 \iff 2.2176x_1 + 1.344x_2 - 4.81 = 0$$

The margin is the distance between any support vector and the boundary line. As was shown above, this distance can be computed as follows:

$$margin = \frac{1}{\|\mathbf{w}\|_2} = \frac{1}{\sqrt{2.2176^2 + 1.344^2}} = 0.386$$

---

e) Using the coefficient formulation, how does this SVM classify $\mathbf{x} = \begin{pmatrix} 1 & 3 \end{pmatrix}^T$.

Solution:

$$\begin{aligned} o(\mathbf{x}) &= sgn\left(\sum_{i=1}^{2} t^{(i)}\alpha_i \mathbf{x}^T\mathbf{x}^{(i)} + b\right) \\ &= sgn\left((+1)(3.36)\begin{pmatrix} 1 & 3 \end{pmatrix}\begin{pmatrix} 2.07 \\ 0.91 \end{pmatrix} + (-1)(3.36)\begin{pmatrix} 1 & 3 \end{pmatrix}\begin{pmatrix} 1.41 \\ 0.51 \end{pmatrix} - 4.81\right) \\ &= sgn(1.44) \\ &= +1 \end{aligned}$$

---

f) What about with the weight vector?.

Solution:

$$\begin{aligned} o(\mathbf{x}) &= sgn\left(\mathbf{w}^T\mathbf{x} + b\right) \\ &= sgn\left(\begin{pmatrix} 2.2176 \\ 1.344 \end{pmatrix}^T\begin{pmatrix} 1 \\ 3 \end{pmatrix} - 4.81\right) \\ &= sgn(1.44) \\ &= +1 \end{aligned}$$

---

3) Consider the training set:

$$\mathbf{X} = \begin{pmatrix} 0.68 & -0.25 & 0.83 & 1.44 & 1.14 & -2.19 & -2.47 & -1.19 & -2.3 & -0.94 \\ 1.26 & -0.51 & -0.32 & 1.04 & 3.21 & 5.1 & 3.76 & 6.39 & 3.48 & 5.04 \end{pmatrix}$$

With targets:

$$\mathbf{t} = \begin{pmatrix} +1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

After running an SVM we get the weight vector $w = \begin{pmatrix} 0.544 & -0.474 \end{pmatrix}^T$ and bias $b = 1.902$.

a) What are the support vectors?

Solution:
We know that, for any support vector $\mathbf{x}^{(i)}$ with target $t^{(i)}$, we have that:

$$\mathbf{w}^T\mathbf{x}^{(i)} + b = t^{(i)}$$

So, all we need to do is check this condition for every point:

– For $\mathbf{x}^{(1)}$: $\begin{pmatrix} 0.544 & -0.474 \end{pmatrix}\begin{pmatrix} 0.68 \\ 1.26 \end{pmatrix} + 1.902 = 1.67 \neq +1$, so it is not a support vector.
– For $\mathbf{x}^{(2)}$: $\begin{pmatrix} 0.544 & -0.474 \end{pmatrix}\begin{pmatrix} -0.25 \\ -0.51 \end{pmatrix} + 1.902 = 2.01 \neq +1$, so it is not a support vector.
– For $\mathbf{x}^{(3)}$: $\begin{pmatrix} 0.544 & -0.474 \end{pmatrix}\begin{pmatrix} 0.83 \\ -0.32 \end{pmatrix} + 1.902 = 2.51 \neq +1$, so it is not a support vector.
– For $\mathbf{x}^{(4)}$: $\begin{pmatrix} 0.544 & -0.474 \end{pmatrix}\begin{pmatrix} 1.44 \\ 1.04 \end{pmatrix} + 1.902 = 2.19 \neq +1$, so it is not a support vector.
– For $\mathbf{x}^{(5)}$: $\begin{pmatrix} 0.544 & -0.474 \end{pmatrix}\begin{pmatrix} 1.14 \\ 3.21 \end{pmatrix} + 1.902 = 1 = +1$, so it is a support vector.
– For $\mathbf{x}^{(6)}$: $\begin{pmatrix} 0.544 & -0.474 \end{pmatrix}\begin{pmatrix} -2.19 \\ 5.1 \end{pmatrix} + 1.902 = -1.71 \neq -1$, so it is not a support vector.
– For $\mathbf{x}^{(7)}$: $\begin{pmatrix} 0.544 & -0.474 \end{pmatrix}\begin{pmatrix} -2.47 \\ 3.76 \end{pmatrix} + 1.902 = -1.22 \neq -1$, so it is not a support vector.
– For $\mathbf{x}^{(8)}$: $\begin{pmatrix} 0.544 & -0.474 \end{pmatrix}\begin{pmatrix} -1.19 \\ 6.39 \end{pmatrix} + 1.902 = -1.78 \neq -1$, so it is not a support vector.
– For $\mathbf{x}^{(9)}$: $\begin{pmatrix} 0.544 & -0.474 \end{pmatrix}\begin{pmatrix} -2.3 \\ 3.48 \end{pmatrix} + 1.902 = -1 = -1$, so it is a support vector.
– For $\mathbf{x}^{(10)}$: $\begin{pmatrix} 0.544 & -0.474 \end{pmatrix}\begin{pmatrix} -0.94 \\ 5.04 \end{pmatrix} + 1.902 = -1 = -1$, so it is a support vector.

So, we have three support vectors: $\mathbf{x}^{(5)}$, $\mathbf{x}^{(9)}$ and $\mathbf{x}^{(10)}$.

---

b) What are the dual coefficients α_i?

Solution:

Looking at the classification function:

$$o(\mathbf{x}) = sgn\left(\mathbf{w}^T\mathbf{x} + b\right)$$

Which can be written in terms of the coefficients as follows:

$$o(\mathbf{x}) = sgn\left(\sum_{i=1}^{10} t^{(i)}\alpha_i\mathbf{x}^T\mathbf{x}^{(i)} + b\right)$$

So, for a support vector $\mathbf{x}$ with target t, we must have:

$$\sum_{i=1}^{10} t^{(i)}\alpha_i\mathbf{x}^T\mathbf{x}^{(i)} + b = t$$

For the first support vector:

$$\begin{aligned}
\sum_{i=1}^{10} t^{(i)}\alpha_i\mathbf{x}^T\mathbf{x}^{(i)} + b &= t\\
t^{(5)}\alpha_5\mathbf{x}^T\mathbf{x}^{(5)} + t^{(9)}\alpha_9\mathbf{x}^T\mathbf{x}^{(9)} + t^{(10)}\alpha_{10}\mathbf{x}^T\mathbf{x}^{(10)} + 1.902 &= +1\\
\alpha_5\left(1.14\ 3.21\right)\begin{pmatrix}1.14\\3.21\end{pmatrix} - \alpha_9\left(1.14\ 3.21\right)\begin{pmatrix}-2.3\\3.48\end{pmatrix} - \alpha_{10}\left(1.14\ 3.21\right)\begin{pmatrix}-0.94\\5.04\end{pmatrix} + 1.902 &= +1\\
11.6037\alpha_5 - 8.5488\alpha_9 - 15.1068\alpha_{10} + 1.902 &= +1\\
11.6037\alpha_5 - 8.5488\alpha_9 - 15.1068\alpha_{10} &= -0.902
\end{aligned}$$

For the second support vector:

$$\begin{aligned}
\sum_{i=1}^{10} t^{(i)}\alpha_i\mathbf{x}^T\mathbf{x}^{(i)} + b &= t\\
t^{(5)}\alpha_5\mathbf{x}^T\mathbf{x}^{(5)} + t^{(9)}\alpha_9\mathbf{x}^T\mathbf{x}^{(9)} + t^{(10)}\alpha_{10}\mathbf{x}^T\mathbf{x}^{(10)} + 1.902 &= -1\\
\alpha_5\left(-2.3\ 3.48\right)\begin{pmatrix}1.14\\3.21\end{pmatrix} - \alpha_9\left(-2.3\ 3.48\right)\begin{pmatrix}-2.3\\3.48\end{pmatrix} - \alpha_{10}\left(-2.3\ 3.48\right)\begin{pmatrix}-0.94\\5.04\end{pmatrix} + 1.902 &= -1\\
8.5488\alpha_5 - 17.4004\alpha_9 - 19.7012\alpha_{10} + 1.902 &= -1\\
8.5488\alpha_5 - 17.4004\alpha_9 - 19.7012\alpha_{10} &= -2.902
\end{aligned}$$

For the third support vector:

$$\begin{aligned}
\sum_{i=1}^{10} t^{(i)}\alpha_i\mathbf{x}^T\mathbf{x}^{(i)} + b &= t\\
t^{(5)}\alpha_5\mathbf{x}^T\mathbf{x}^{(5)} + t^{(9)}\alpha_9\mathbf{x}^T\mathbf{x}^{(9)} + t^{(10)}\alpha_{10}\mathbf{x}^T\mathbf{x}^{(10)} + 1.902 &= -1\\
\alpha_5\left(-0.94\ 5.04\right)\begin{pmatrix}1.14\\3.21\end{pmatrix} - \alpha_9\left(-0.94\ 5.04\right)\begin{pmatrix}-2.3\\3.48\end{pmatrix} - \alpha_{10}\left(-0.94\ 5.04\right)\begin{pmatrix}-0.94\\5.04\end{pmatrix} + 1.902 &= -1\\
15.1068\alpha_5 - 19.7012\alpha_9 - 26.2852\alpha_{10} + 1.902 &= -1\\
15.1068\alpha_5 - 19.7012\alpha_9 - 26.2852\alpha_{10} &= -2.902
\end{aligned}$$

Combining the three equations in a system and solving it we get:

$$\alpha_5 = 0.2606$$

$$\alpha_9 = 0.0016$$

$$\alpha_{10} = 0.2600$$

---

c) What is the value of the margin?

Solution:

The margin is the distance between any support vector and the boundary line. As was shown above, this distance can be computed as follows:

$$margin = \frac{1}{\|\mathbf{w}\|_2} = \frac{1}{\sqrt{0.544^2 + -0.474^2}} = 1.385$$

---

3 Thinking Questions

a) Try to show the VC dimension of a perceptron without bias.

b) Think about how learning theory can help you in practice. Namely, think about how one can use it to choose a classifier and to make decisions on regularization parameters.

c) Think about the relation between the Bias-Variance decomposition and the VC dimension.

d) Think about the VC dimension of an SVM. Will it be closer to the dimension of the weight vector? Will it be related to the number of support vectors?

e) Think about how the choice for SVM kernel related to the feature transformations we applied with linear regression.

e) How do you think the kernel related to the use of hidden layers in MLPs?

f) Is an MLP with linear hidden layers able to separate nonlinear problems?

e) What about an SVM with a linear kernel?