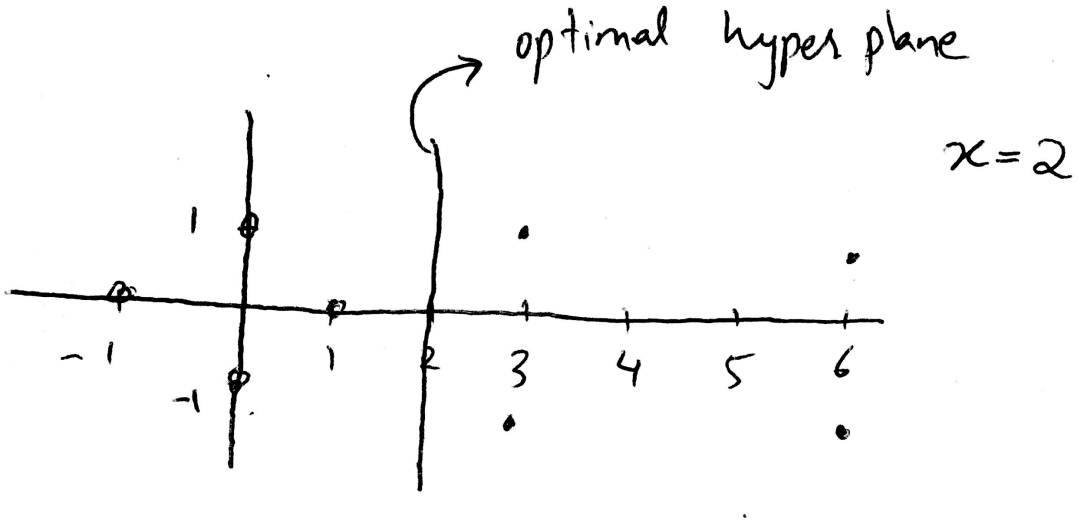
1.1)




Positive points are represented by filled circles, to the right of the hyperplane, and negative by empty circles to the left.

Optimal hyperplane lies at x=2, and margin = 1.
Points acting as the support vectors are, (3,1), (3,-1) and (1,0)

1.2) Yes, the point (1.8 ,1) label -1, is correctly classified by the optimal hyperplane in the above SVM.

1.3) For the vanilla perceptron, there are no margin guarantees due to the lack of a regularization term. For 0% error on the training dataset, the perceptron can be any line lying in the region between points (1,0) and (3,0), such that it classifies the points correctly. For example, x=1.5 can be the equation of a perceptron (or weights 1.5, 0) which correctly classifies the training data with 0% error, however it will classify the point x=[1.9999,1] with label -1 wrongly.

2.1) Given a hypothesis space of n dimensions, such that

$x \in \{0,1\}^n$ and each point consists of features, d1, d2, d3, d4….dn

Let the true function be a 2-dnf conjunction such that

$$f(x) = (d1 \bigwedge \neg d2) \bigvee (d3 \bigwedge \neg d4) \ldots$$

We can implement a transformation on x such that

$$\phi(x) = (d1 \bigwedge \neg d2\ ,\ d3 \bigwedge \neg d4, \ldots)$$

A transformation to this space can be linearly separated by a weight vector of weights 1,1,1… and bias 0.

This logic can be extended easily to a k-DNF where each term of the transformation would be the conjunction of k-literals from the d dimensions, as demonstrated above.

2.2)We wish to compute the function

$$K(x_1, x_2) = \phi(x_1)^T \phi(x_2) = \sum_{c \in C} c(x_1)c(x_2)$$

Where C is the set of all conjunctions of exactly K literals.

Looking at the third term, we can easily see that the summation term will be affected only when $c(x_1)=c(x_2)=1$.
We can thus conclude, that c only contains feature values that are the same for both $x_1$ and $x_2$. Thus we pick k elements from the set of above values, and find their sum.

$$K(x_1, x_2) = \sum same(x, z)\ Choose\ k$$

It would take linear time to calculate the set of same features. Thus the summation can be easily and efficiently calculated without having to find the dot product.

2.3) In perceptron algorithm,
let true label = $y_i$, and predicted label=$y_i'$
Condition for update:
if $y^*(w^T x)>=0$
$w \leftarrow w - y_i(\phi(x_i))$

assuming $w_0= 0$, and x belonging to set of Mistakes, M
$w_1 = - y_1(\phi(x_1))$
$w_2 = - y_1(\phi(x_1)) + - y_2(\phi(x_2))$
$w_3 = - y_1(\phi(x_1)) + - y_2(\phi(x_2)) + - y_3(\phi(x_3))$
$w_n = - y_1(\phi(x_1)) + - y_2(\phi(x_2)) + - y_3(\phi(x_3)) + \ldots + - y_n(\phi(x_n))$
Thus w can simply be expressed with a new transformation which consists of the negative sign

$\sum_{(x_i, y_i) \in M} y_i \phi(x_i)$ where $M=\{(x_i,y_i)\}$ is the set of points where the algorithm makes mistakes.

2.4) Given classification step
$y = sgn(w^T \phi(x_c))$ *where* $x_c$ *is the current point*
replacing $w^T$ from above,

$$y = sgn(\sum_{(x_i, y_i) \in M} y_i \phi(x_i)^T \phi(x_c))$$

replacing $(x_i)^T \phi(x_c)$ with $K(x_i, x_c)$ from 2.2) we get

$$y = sgn(\sum_{(x_i, y_i) \in M} y_i K(x_i, x_c))$$

2.5)
Initialize M - mistake vector as 0
For each ($x_c$, $y_c$) in training set,

if $y = sgn(\sum_{(x_i,y_i)\in M} y_i K(x_i, x_c))$ !=$y_c$ where M = Mistake vector

add current point ($x_c$,$y_c$ )to mistake vector M.