Chapter 9: Support Vector Machines

categorical response Y

The *support vector machine* is an approach for classification that was developed in the computer science community in the 1990s and has grown in popularity.

SVMs perform well in a variety of settings

often considered one of the best "out of the box" classifiers.

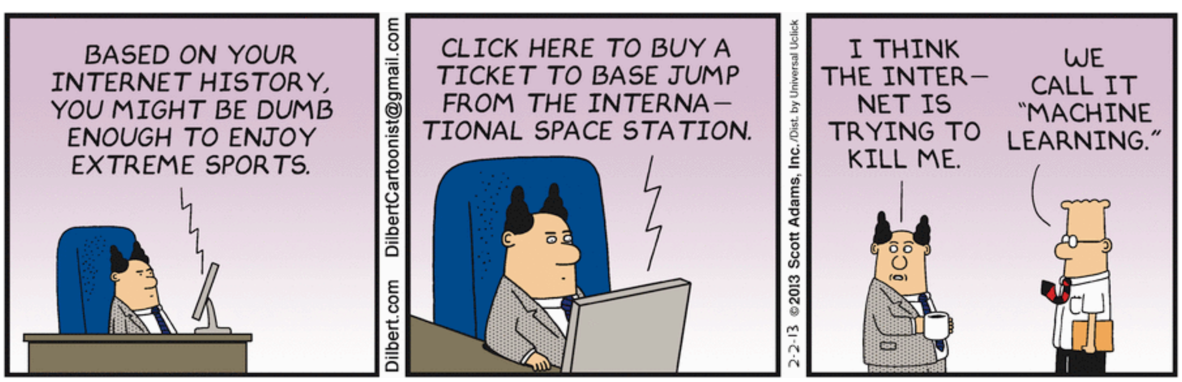
The support vector machine is a generalization of a simple and intuitive classifier called the *maximal margin classifier*.

Sometimes all 3 referred to as SVM.

$\boxed{\text{maximal margin classifier}} \xrightarrow{\text{generalization: allow imperfect separation of training data}} \boxed{\text{support vector classifier}} \xrightarrow{\text{generalization: nonlinear decision boundary}} \boxed{\text{support vector machine}}$

Support vector machines are intended for binary classification, but there are extensions for more than two classes.

categorical response w/ only 2 classes.

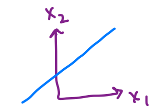


Credit: https://dilbert.com/strip/2013-02-02 based on a hyperplane separator.

1 Maximal Margin Classifier

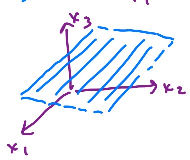
extension of euclidean space.

In p-dimensional space, a *hyperplane* is a flat affine subspace of dimension $p - 1$.



e.g. In 2 dimensions, a hyperplane is a flat 1 dimensional subspace – a line.

In 3 dimensions, a hyperplane is a flat 2 dimensional subspace – a plane



⋮

In $p > 3$ dimensions, harder to conceptualize, but still a flat $p-1$ dim. subspace.

The mathematical definition of a hyperplane is quite simple,

parameters

In 2 dimensions, a hyperplane is defined by $\underline{\beta_0} + \underline{\beta_1} X_1 + \underline{\beta_2} X_2 = 0$

i.e. any $\underline{X} = (x_1, x_2)$ for which this equation holds lies on the hyperplane.

Note this is just the equation of a line.

This can be easily extended to the p-dimensional setting.

$\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p = 0$ defines a p-dim hyperplane.

i.e. any $\underline{x} = (x_1, \dots, x_p)$ for which this equation holds lies on the hyperplane.

We can think of a hyperplane as dividing p-dimensional space into two halves.

If: $\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p > 0$ then $\underline{X} = (x_1, \dots, x_p)$ lies on one side of the hyperplane.

$\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p < 0$ then $\underline{X}$ lies on the other side of the hyperplane.

You can determine which side of the hyperplane by just determining the sign of

$\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p$

1.1 Classificaton Using a Separating Hyperplane

Suppose that we have a $n \times p$ data matrix $\boldsymbol{X}$ that consists of n training observations in p-dimensional space.

$$\underline{x}_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, \underline{x}_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

training observations.

and that these observations fall into two classes.

$y_1, \dots, y_n \in \{-1, 1\}$

where -1 represents one class
1 represents the other class.

We also have a test observation.

p-vector of observed features:

$$\underline{x}^* = (x_1^*, \dots, x_p^*)^T$$

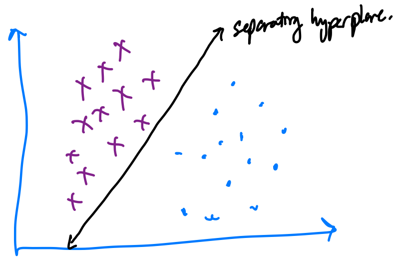
Our Goal: Develop a classifier based on training data that will correctly classify the test observation based on feature measurements.

We have already used many approaches:

- logistic regression
- KNN
- RF, boosting, bagging
- trees
- LDA, QDA

We will see a new approach using a <u>separating hyperplane</u>.

Suppose it is possible to construct a hyperplane that separates the training observations perfectly according to their class labels.



Then a separating hyperplane has the property that

$\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} > 0$ if $y_i = 1$ and
$\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} < 0$ if $y_i = -1$ for $i = 1, \dots, n$.

$\Longleftrightarrow$

$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) > 0$ for $i = 1, \dots, n$.

If a separating hyperplane exists, we can use it to construct a very natural classifier:

a test observation is assigned to a class depending on which side of the hyperplane it is located.

That is, we classify the test observation x^* based on the sign of
$f(x^*) = \beta_0 + \beta_1 x_1^* + \dots + \beta_p x_p^*$.

If $f(x^*) > 0$ assign x^* to class 1

If $f(x^*) < 0$ assign x^* to class -1.

We can also use the magnitude of $f(x^*)$.

If $f(x^*)$ is is far from zero, this means x^* lies far from the hyperplane

$\Rightarrow$ we can be confident about our class assignment for x^*

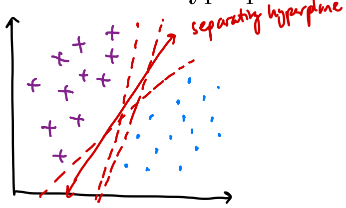
If $f(x^*)$ is close to zero, it is located near the hyperplane

$\Rightarrow$ we are less sure about class assignment.

Note: a classifier based on a hyperplane leads to a linear decision boundary.

1.2 Maximal Margin Classifier

If our data can be perfectly separated using a hyperplane, then there will exist an infinite number of such hyperplanes.

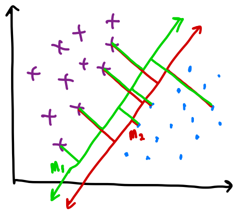


a given separating hyperplane can be shifted a tiny bit up or down or rotated without coming into contact w/ any observations. ⟹ Which one to use for our classifier?

A natural choice for which hyperplane to use is the *maximal margin hyperplane* (aka the *optimal separating hyperplane*), which is the hyperplane that is farthest from the training observations.

- We compute the perpendicular distance from each observation to a given separating hyperplane
- the smallest distance is called the margin.

The maximal margin hyperplane is the hyperplane w/ the largest margin, i.e. farthest from all training points,



$m_1 > m_2$

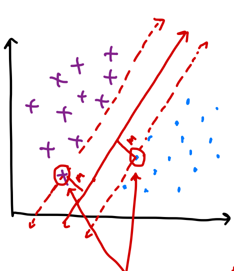
⟹ larger margin

⟹ green is my preferred hyperplane.

We can then classify a test observation based on which side of the maximal margin hyperplane it lies – this is the *maximal margin classifier.*

- hopefully a large margin in training data will lead to a large margin on test data
  ⟹ classify a test data set correctly.

- When p is large, overfitting can occur.



These two points are known as the support vectors because they are p-dim vectors that "support" the hyperplane.

these two points are equidistant from the maximal margin hyperplane.

i.e. If these 2 points move, the maximal margin hyperplane would move as well.

NOTE: The maximal margin hyperplane only depends on the support vectors! (a small # of points!)
The rest of the points can move and it doesn't matter.

We now need to consider the task of constructing the maximal margin hyperplane based on a set of n training observations and associated class labels.

$\underline{x}_1, \ldots, \underline{x}_n \in \mathbb{R}^p$ $\qquad$ $y_1, \ldots, y_n \in \{-1, 1\}$.

The maximal margin hyperplane is the solution to the optimization problem:

① $\underset{\beta_0, \beta_1, \ldots, \beta_p, M}{\text{Maximize}} \; M$ ← margin

subject to $\sum_{j=1}^{p} \beta_j^2 = 1$ ②

$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M \quad \forall\, i = 1, \ldots, n$ ③

③ means each observation in training data set will be on the correct side of the hyperplane ($M \geq 0$) with some cushion if ($M > 0$).

② ensures $y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip})$ is perp. distance to the hyperplane and ③ means the point x_i is atleast M away $\Rightarrow$ defines M as the margin.

① chooses $\beta_0, \ldots, \beta_p, M$ to maximize the margin.

$\Rightarrow$ maximal margin hyperplane!

This problem can be solved efficiently, but the details are outside the scope of this course.

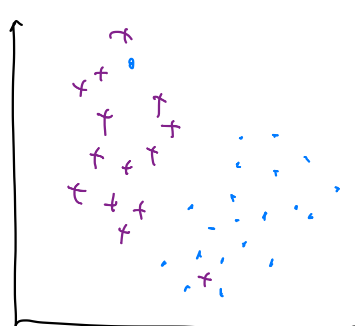
↳ we'll talk more later....

What happens when no separating hyperplane exists?

$\Rightarrow$ no maximal margin hyperplane!

We can develop a hyperplane that almost separates the classes

↳ a "soft margin"

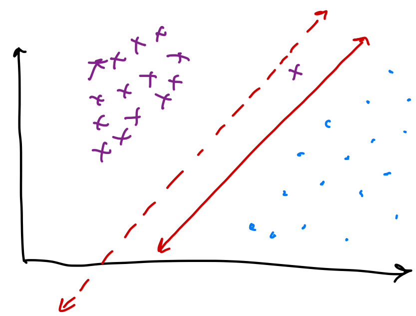


We can't draw a hyperplane to separate perfectly!

2 Support Vector Classifiers

It's not always possible to separate training observations by a hyperplane. In fact, even if we can use a hyperplane to perfectly separate our training observations, it may not be desirable.

A classifier based on a perfectly separating (maximal margin) hyperplane can lead to oversensitivity to individual observations (high variability).



A single data point can have a large effect on the hyperplane (w/ smaller margin).

We might be willing to consider a classifier based on a hyperplane that does *not perfectly* separate the two classes in the interest of

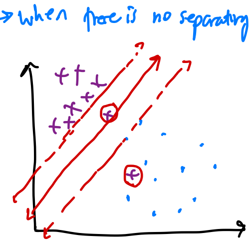
- greater robustness to individual observations
- proper classification of most of the training observations.

i.e. it might be worthwhile to misclassify a few observations in training data set to do a better job classifying a test data set.

"soft margin classifier"

The *support vector classifier* does this by finding the largest possible margin between classes, but allowing some points to be on the "wrong" side of the margin, or even on the "wrong" side of the hyperplane.

↳ when there is no separating hyperplane this is inevitable.



wrong side of margin or hyperplane. That's okay!

The support vector classifier classifies a test observation depending on which side of the hyperplane it lies. The hyperplane is chosen to correctly separate **most** of the training observations.

Solution to the following optimization problem:

$$\underset{\beta_0,\beta_1,\dots,\beta_p,\varepsilon_1,\dots,\varepsilon_n,M}{\text{maximize}} \quad M$$

subject to

$$\sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M(1-\varepsilon_i)$$

$$\varepsilon_i \geq 0, \quad \sum_{i=1}^{n} \varepsilon_i \leq \underline{C}$$

"slack variables" allow observations to be on the wrong side of margin (or hyperplane)

nonnegative tuning parameter (budget for how wrong we are willing to be on training data).

Once we have solved this optimization problem, we classify x^* as before by determining which side of the hyperplane it lies.

classify x^* based on sign of $f(x^*) = \beta_0 + \beta_1 x_1^* + \dots + \beta_p x_p^*$.

ϵ_i – tells us where the observation lies relative to hyperplane and margin.

If $\varepsilon_i = 0 \Rightarrow$ obs. on correct side of margin.

If $\varepsilon_i > 0 \Rightarrow$ obs. on wrong side of margin

If $\varepsilon_i > 1 \Rightarrow$ obs. on wrong side of hyperplane

C – tuning parameter, bounds the sum of ε_i's $\Rightarrow$ determines # and severity of violations we will allow

think of C as a budget for the amount of violations.

If $C = 0 \Rightarrow$ no budget for violations $\Rightarrow \varepsilon_1 = \dots = \varepsilon_n = 0 \Rightarrow$ SV classifier = maximal margin classifier.

If $C > 0 \Rightarrow$ no more than C obs. can be on the wrong side of the hyperplane because $\varepsilon_i > 1$ and $\sum_{i=1}^{n} \varepsilon_i \leq C$.

$$\left.\begin{array}{l}\text{small } C \Rightarrow \text{narrow margins} \\ \text{large } C \Rightarrow \text{wider margin, allow for more violations}\end{array}\right\} \text{controls the bias-variance trade-off.} \Rightarrow \text{choose } C \text{ by cross-validation!}$$

The optimization problem has a very interesting property.

only observations on the margin or violate the margin (or hyperplane) affect the hyperplane!
⇒ the classifier.

i.e. observations that lie on correct side of margin do not affect the support vector classifier!

Observations that lie directly on the margin or on the wrong side of the margin or hyperplane are called *support vectors*.

These observations do affect the classifier.

The fact that only support vectors affect the classifier is in line with our assertion that C controls the bias-variance tradeoff.

When C large ⇒ margin is wide, many observations violating margin or hyperplane
⇒ many support vectors
⇒ many observations used to determine the hyperplane
⇒ low variance but potentially high bias.

When C small ⇒ fewer support vectors
⇒ low bias but high variance.

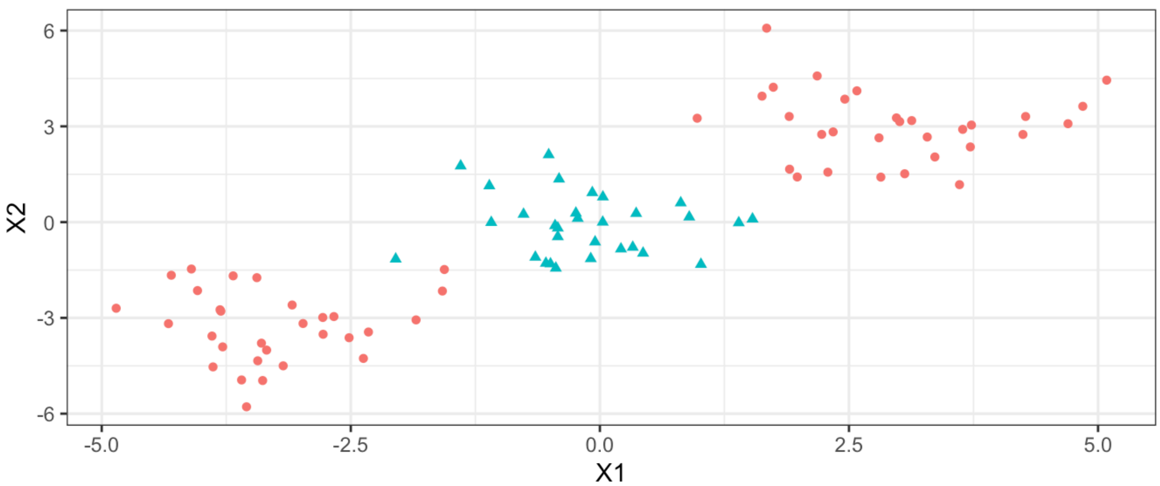
Because the support vector classifier's decision rule is based only on a potentially small subset of the training observations means that it is robust to the behavior of observations far away from the hyperplane.

distinct from behavior of other methods

e.g. LDA depends on mean of observations in each class.

3 Support Vector Machines

The support vector classifier is a natural approach for classification in the two-class setting...



We've seen ways to handle non-linear classification boundaries before.

In the case of the support vector classifier, we could address the problem of possible non-linear boundaries between classes by enlarging the feature space.

Then our optimization problem would become

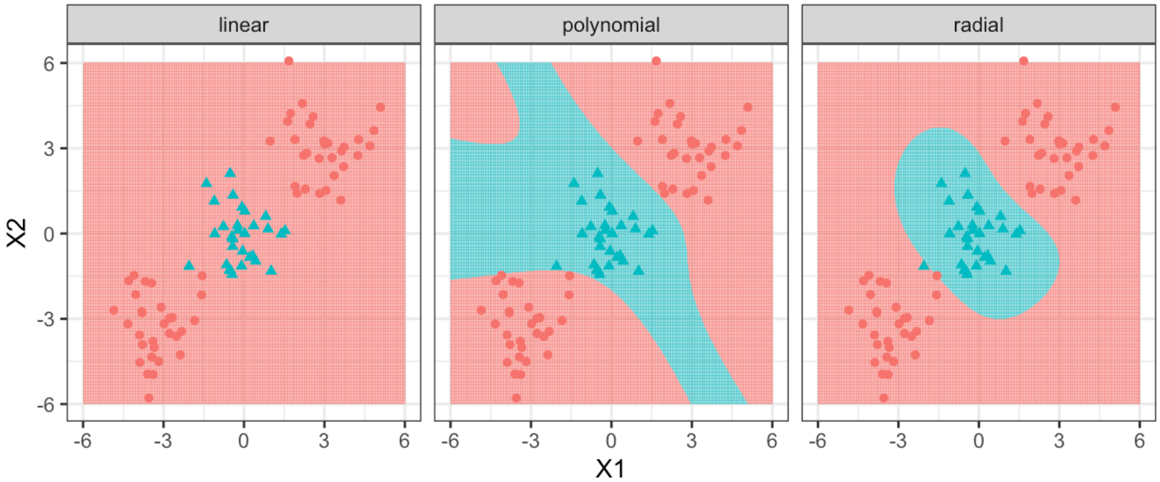
The *support vector machine* allows us to enlarge the feature space used by the support classifier in a way that leads to efficient computation.

It turns out that the solution to the support vector classification optimization problem involves only *inner products* of the observations (instead of the observations themselves).

It can be shown that

-
-
-

Now suppose every time the inner product shows up in the SVM representation above, we replaced it with a generalization.


linear
polynomial
radial
6
3
0
-3
-6
X2
-6
-3
0
3
6
X1

4 SVMs with More than Two Classes

So far we have been limited to the case of binary classification. How can we exted SVMs to the more general case with some arbitrary number of classes?

Suppose we would like to perform classification using SVMs and there are $K > 2$ classes.

One-Versus-One

One-Versus-All