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

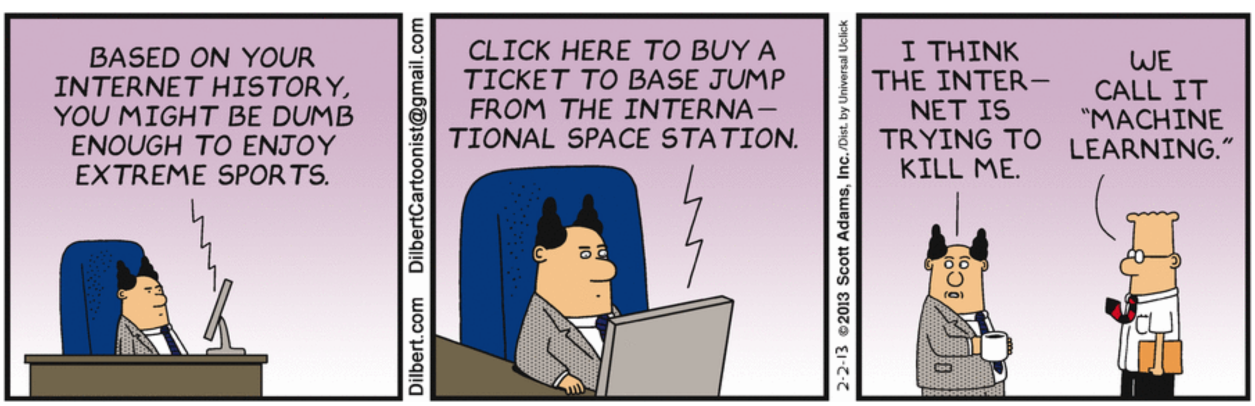
The support vector machine is a generalization of a simple and intuitive classifier called the *maximal margin classifier*.

maximal margin classifier → generalization: allow imperfect separation of training data → support vector classifier → generalization: allow nonlinear decision boundary → support vector machine.

sometimes all 3 referred to as SVM.

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

categorical response w/ only 2 classes.

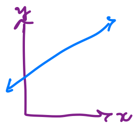


Credit: https://dilbert.com/strip/2013-02-02 based on a hyperplane separator

1 Maximal Margin Classifier

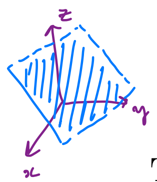
extension of euclidean space.

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



e.g. In 2 dimension, 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 $p-1$ dimen. 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 $X = (X_1, X_2)$ for which this equation holds lies on the hyperplane.

Note this is just the equation for a line.

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

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

i.e. any $X = (X_1, \ldots, X_p)$ for which this equation holds lies on the hyperplane.

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

If $\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p > 0$ then X lies on one side of the hyperplane

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

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

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

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.

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

training observations

and that these observations fall into two classes.

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

where -1 represets one class

1 represents the other class.

We also have a test observation.

p-vector of obsered features

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

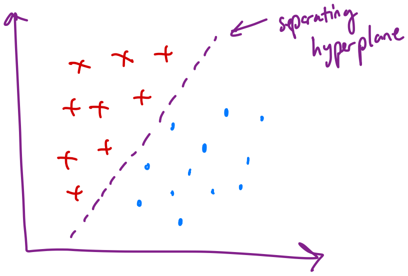
Our Goal: Develop a classifier based on training data that will correctly classify the test observation based on feature measurements.

We have already used many approaches:

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

We will see a new approach using a separating hyperplane.

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

$$\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} > 0 \quad \text{if } y_i = 1 \quad \text{and}$$

$$\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} < 0 \quad \text{if } y_i = -1$$

$$\iff$$

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

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

a test observation is assigned 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 $f(x^*) = \beta_0 + \beta_1 x_1^* + \cdots + \beta_p x_p^*$.

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 far from zero, this means x^* lies far from the hyperplane

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

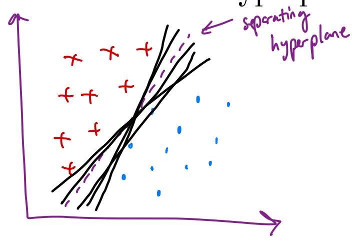
If $f(x^*)$ is close to zero, it is located near the hyperplane

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

Note: a classifier based on a separating 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.

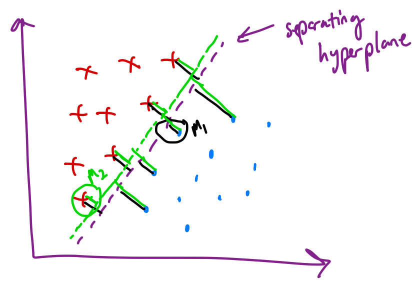


a given separating hyperplane can usually be shifted a tiny bit up or down or rotated without coming into contact w/ any observations. ⟹ which one to use for our classifier.

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 known as the margin.

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



$M_1 > M_2$

⟹ larger margin

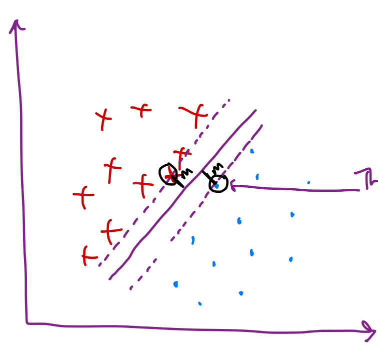
⟹ purple is my prefered hyperplane.

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 on the training data will lead to a large margin on test data

  ⟹ classify test data correctly.

- When p is large, overfitting does occur.



These two points are equidistant from the maximal margin hyperplane.

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

$x_1, \dots, x_n \in \mathbb{R}^p$ $\qquad$ $y_1, \dots, y_n \in \{-1, 1\}$.

The maximal margin hyperplane is the solution to the optimization problem

① $\underset{\beta_0, \dots, \beta_p, M}{\text{Maximize}}\ M$ ← margin

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

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

③ means each observation will be on the correct side of hyperplane ($M \geq 0$) w/ some cushion (if $M > 0$).

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

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

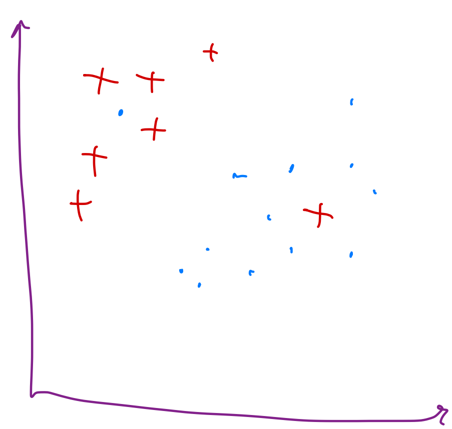
$\Rightarrow$ maximal margin hyperplane!

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

↳ we'll talk a little bit more later.

What happens when no separating hyperplane exists?

$\Rightarrow$ no maximal margin hyperplane!



We can't draw a hyperplane to separate these perfectly!

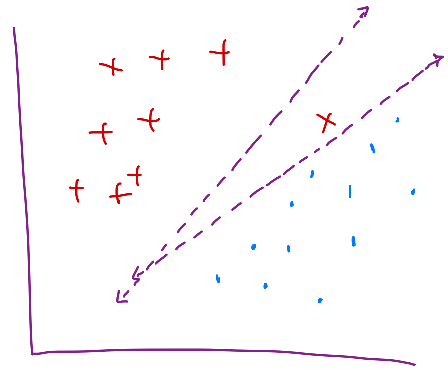
We can develop a hyperplane that almost separates the classes — a "soft margin"

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 hyperplane will necessarily perfectly classify all training obs.

This can lead to oversensitivity to individual observations.



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

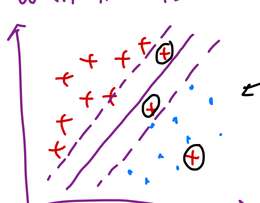
- proper classification of most of the training observations

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

→ Sometimes called "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:

$$\underset{\beta_0,\beta_1,\dots,\beta_p,\varepsilon_1,\dots,\varepsilon_n,M}{\text{maximize}} \; M \leftarrow \text{margin}$$

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 the 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 the margin.

$\varepsilon_i > 0 \Rightarrow$ obs. on wrong side of margin (violated margin)

$\varepsilon_i > 1 \Rightarrow$ 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 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 $\sum_{i=1}^{n} \varepsilon_i \leq C$

small $C \Rightarrow$ narrow margins, large $C \Rightarrow$ wider margins, allow for more violations — controls bias-variance tradeoff

$\Rightarrow$ choose C by CV.

The optimization problem has a very interesting property.

Only observations on the margin or violate the margin or hyperplane affect the hyperplane!

$\Rightarrow$ the classifier.

i.e. observations that lie on the correct side of the 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 $\Rightarrow$ margin is wide, many observations violating margin or hyperplane

$\Rightarrow$ many support vectors

$\Rightarrow$ many observations used to determine the hyperplane.

$\Rightarrow$ low variance but potentially high bias.

When C small $\Rightarrow$ fewer support vectors

$\Rightarrow$ 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 classifier methods

e.g. LDA depends on mean of a observations within each class & within class covariance matrix.

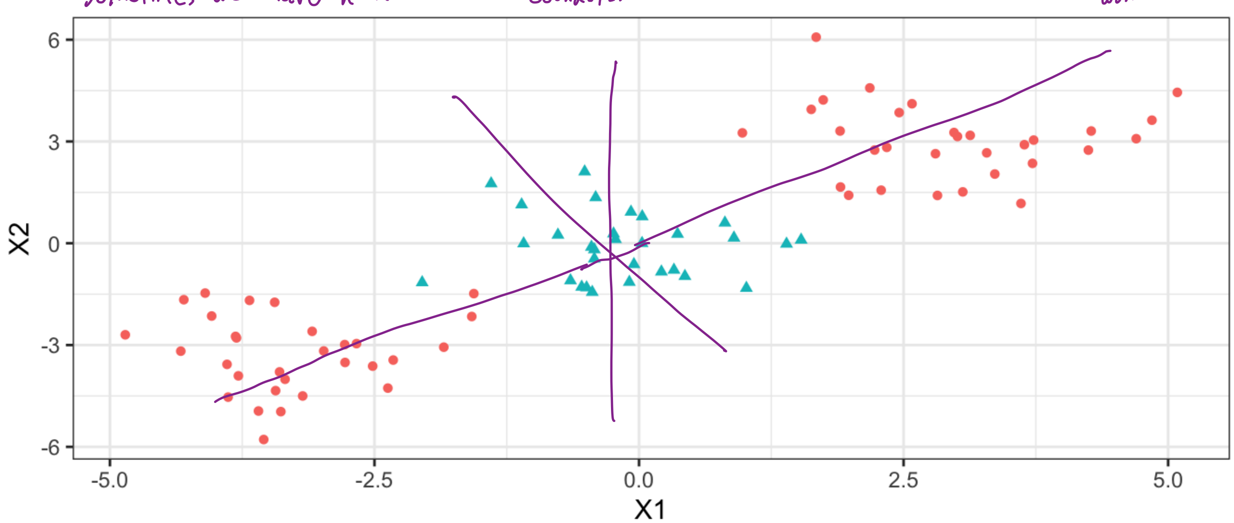
3 Support Vector Machines

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

if the decision boundary is linear!

how to draw a line separating? won't work well.

sometimes we have nonlinear class boundaries



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

bagging, RF, boosting

nonlinear basis function + logistic regression, KNN, QDA

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.

e.g. adding quadratic or cubic terms

instead of fitting SV classifier w/ $X_1, \dots, X_p$

could use $X_1, \dots, X_p, X_1^2, \dots, X_p^2$ etc.

Then our optimization problem would become

$$\underset{\beta_0, \beta_{11}, \beta_{12}, \dots, \beta_{1p}, \beta_{21}, \dots, \beta_{2p}, \varepsilon_1, \dots, \varepsilon_n, M}{\text{Maximize}} \; M$$

subject to

$$\sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{kj}^2 = 1$$

quadratic polynomial leads to nonlinear boundary

$$y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_{1j} x_{ij} + \sum_{j=1}^{p} \beta_{2j} x_{ij}^2 \right) \geq M(1 - \varepsilon_i)$$

$$\sum_{i=1}^{n} \varepsilon_i \leq C$$

Could consider higher order polynomials or other functions.

using "kernel"s

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

Want to enlarge feature space to have non-linear boundary.

computation of support classifier... idea.

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).

inner product $\langle a,b\rangle = \sum_{i=1}^{r} a_i b_i$

inner product of two obs: $\langle x_i, x_{i'}\rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$

It can be shown that

- The (linear) support vector classifier can be written as
  $f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i\rangle$ α_i, $i=1,\dots,n$ additional parameters.
- To estimate $\alpha_1,\dots,\alpha_n$ and β_0 need $\binom{n}{2} = \frac{n(n-1)}{2}$ inner products between all pairs of training observations
- α_i nonzero only for support vectors in the solution!
  ↳ typically less than n

we only need inner products, not observations themselves.

$\Rightarrow$ rewrite $f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i\rangle$, S = indices of support vectors.

Now suppose every time the inner product shows up in the SVM representation above, we replaced it with a generalization. $\langle x, x_i\rangle$

Kernel: $K(x_i, x_{i'})$. (some function).
↳ a function that quantifies similarity of two observations.

e.g. $K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$ results in support vector classifier "linear kernel" bc linear boundary.

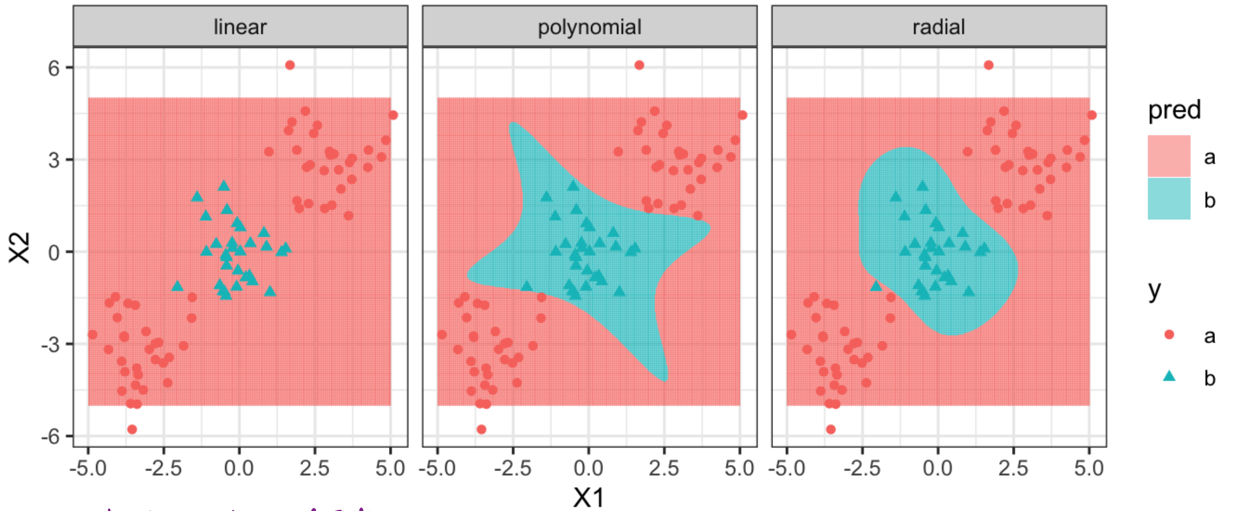
"support vector machine"

$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$ ← pos. integer "polynomial kernel"

$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right)$ "radial kernel"

↑ pos. constant

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i)$$



when $d=1$ in polynomial SVM, same as linear SVM.

$d = 4$

= fitting a support vector classifier in a higher dimensional feature space w/ polynomials of degree d.

$\gamma = 2$

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right)$$

If a test observation x^* that is far from a training observation, then $\sum_{j=1}^{p} (x^*_j - x_{ij})^2$ will be large $\Rightarrow$

$K(x^*, x_i)$ will be very small

$\Rightarrow$ will play no/small role in determining $f(x^*)$.

$\Rightarrow$ training observations far from x^* will play little to no role in prediction of class of x^*.

Why use a kernel instead of enlarging feature space using functions of features?

- computational – only need to compute $K(x_i, x_{i'})$ $\forall$ $\binom{n}{2}$ distinct pairs i, i'
  - don't have to explicitly work in an enlarged space (may be too large to compute hyperplane)
  - radial kernel – enlarged feature space is infinite dimensional!

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?

This is actually not that clear. There is no <u>one</u> obvious way to do this.

Two popular options:

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

One-Versus-One

(1) Construct $\binom{K}{2}$ SVMs each comparing a pair of classes.

(2) Classify a test observation using each of the $\binom{K}{2}$ SVMs

(3) Assign test observation to class it was most frequently assigned to.

One-Versus-All Let x^* be a test observation.

(1) Fit K SVMs comparing each class to remaining $K-1$ classes.

(2) assign x^* to the class for which $\beta_{0K} + \beta_{1K}x_1^* + \dots + \beta_{pK}x_p^*$ is largest.

(results in high level of confidence test observation belongs to K^{th} class over any other).