Support Vector Machines

About the Name...

A Support Vector

A training sample used to define classification boundaries in SVMs

- located near class boundaries

Support Vector Machines

Binary classifiers whose decision boundaries are defined by support vectors

SVMs: Design Principles

Discriminative

Similar to perceptrons, SVMs define linear decision boundaries for two classes directly

- vs. Generative approaches, where decision boundaries defined by estimated posterior probabilities (e.g. LDC, QDC, k-NN)
- Perceptron: decision boundary sensitive to initial weights, choice of η (learning rate), order in which training samples are processed

Maximizing the Margin

Unlike perceptrons, SVMs produce a *unique* boundary between linearly separable classes: the one that maximizes the margin (distance to the decision boundary) for each class

- Often leads to better generalization

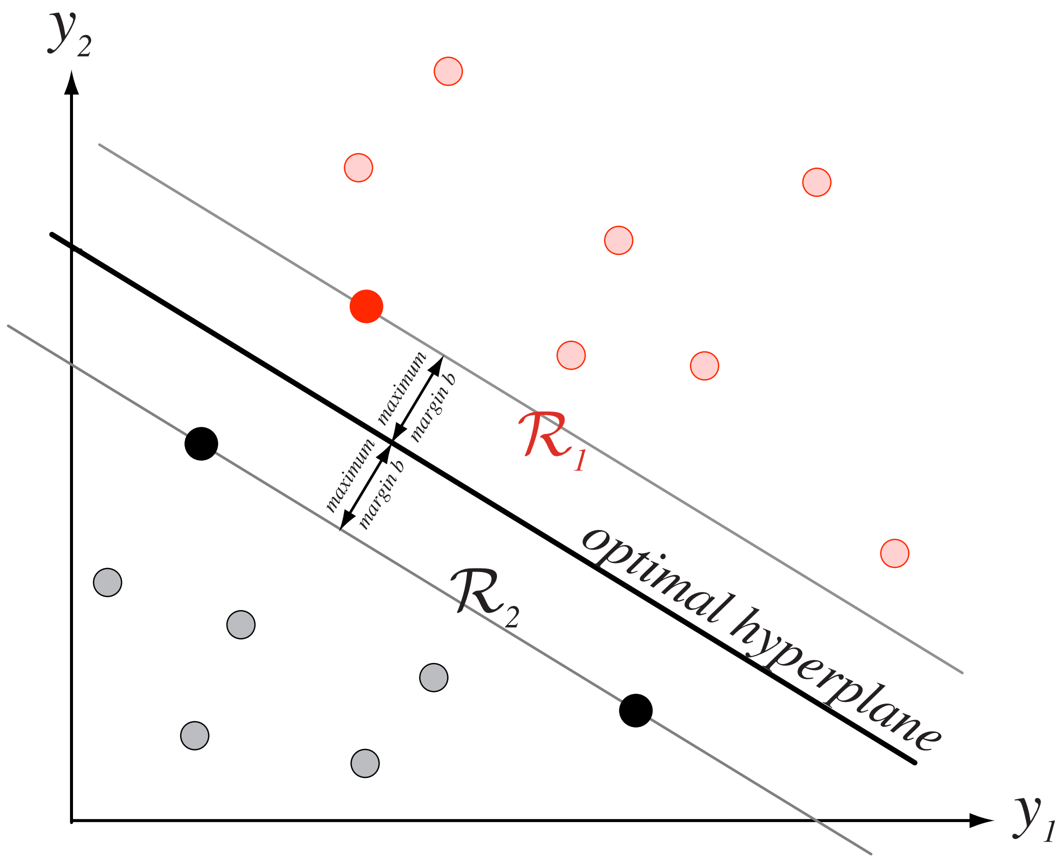


FIGURE 5.19. Training a support vector machine consists of finding the optimal hyperplane, that is, the one with the maximum distance from the nearest training patterns. The support vectors are those (nearest) patterns, a distance *b* from the hyperplane. The three support vectors are shown as solid dots. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons,
'

More on ‘The Margin’

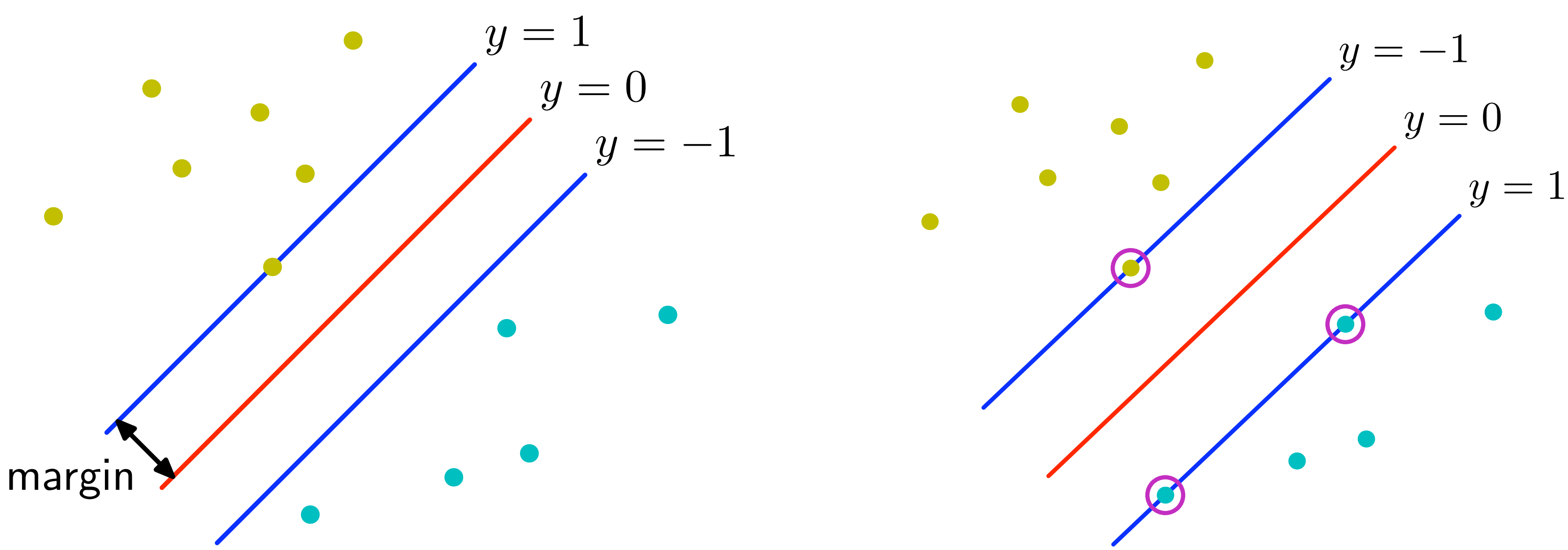
The Margin, *b*

Is the minimum distance of *any* sample to the decision boundary.

Training SVMs

= maximizing the margin, moving the decision boundary as far away from all training samples as possible.

Maximizing the Margin



- At left: a sub-optimal margin
- At right: optimal margin
- y values: for linear function defined by the SVM. For linearly separable data, all training instances correctly classified as -1 or 1 (locations in the margins have y values in (-1,1))

*Bishop, "Pattern Recognition and Machine Learning," p. 327

Binary Classification by SVM

SVMs Define Linear Decision Boundaries

Recall: so do perceptrons, LDCs for two classes, etc.

Classify By the Sign (+/-) of:

N_s: # support vectors x_i (with λ_i > 0)

$$g(x) = w^T x + w_0$$

$$= \sum_{i=1}^{N_s} \lambda_i \; y_i \; x_i^T x + w_0$$

*Here, y_i refers to class (-1 or 1) for instance x_i

where N_s is the number of support vectors, y_i the class of support vector xi (+1 or -1), and λ_i is a weight (Lagrange multiplier) associated with x_i.

Training/Learning for SVMs

Optimization Problem:

$$\max_{\lambda} \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \boxed{\mathbf{x_i}^{\mathbf{T}} \mathbf{x_j}} \right)$$

$$subject\ to \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\forall \mathbf{x_i}, \lambda_i \geq 0$$

Note:

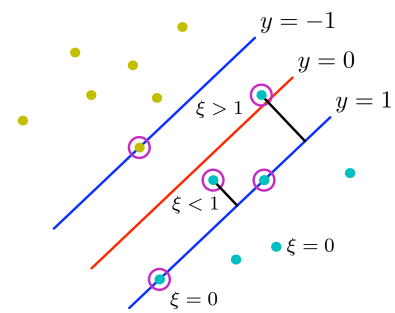
Given a training set, two parameters of g(x) need to be learned/defined: λ_i and w_0

Once λ_i have been obtained, the optimal hyperplane and w_0 may be found

R·I·T

Non-Linearly Separable Classes

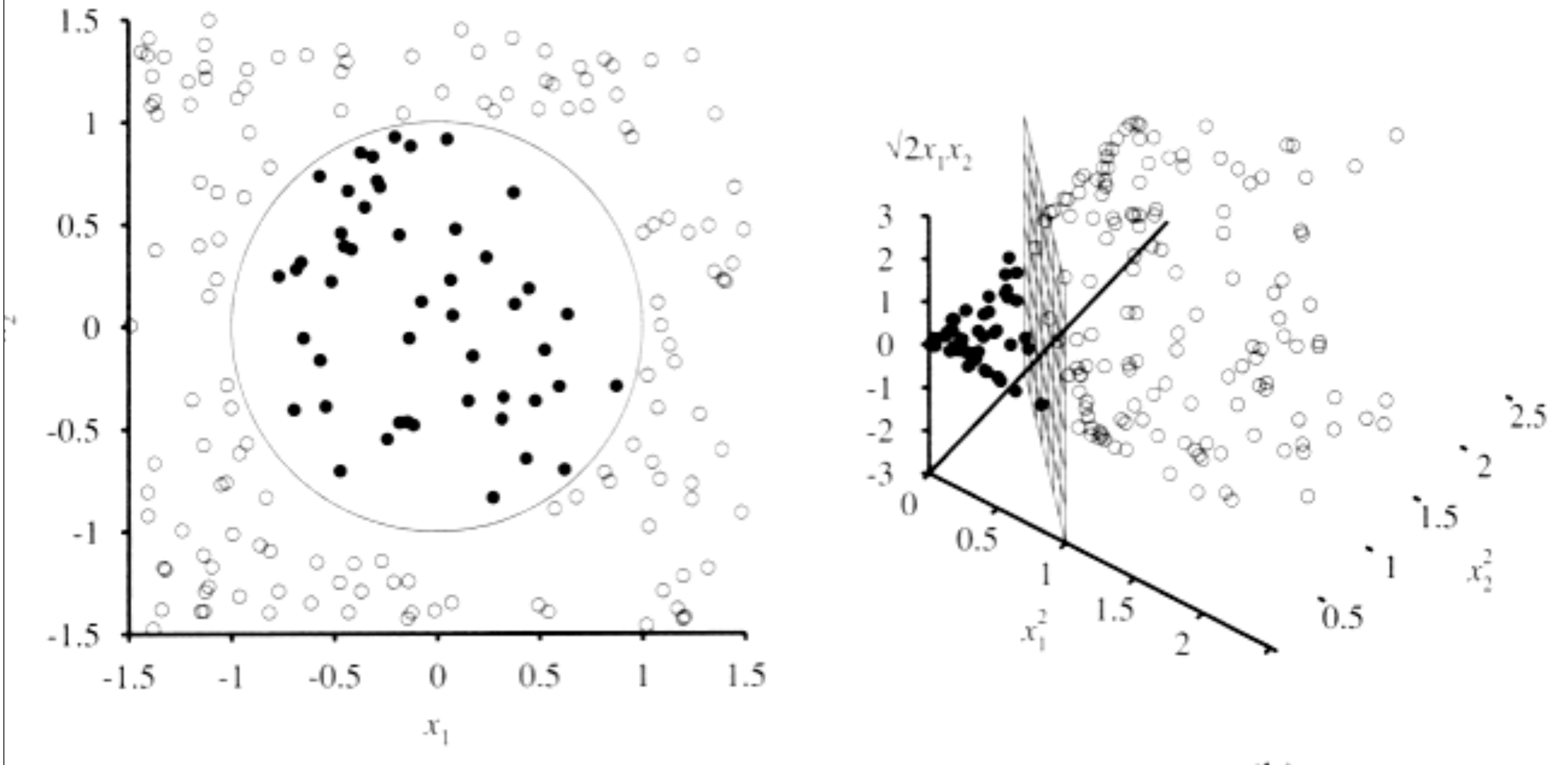
May be handled by using a *soft margin*, in which points may lie, and classification errors may occur (e.g. margin properties defined by tunable parameters for v-SVM).



Often handled by transforming a non-linearly separable feature space into a higher-dimensional one in which classes are linearly separable (the "kernel trick"), and then use a 'plain' SVM for classification.

R·I·T

Example: Restructuring Feature Space
(from Russell & Norvig, 2nd Edition)



Here, mapping is defined by:

$f_1 = x_1^2$ $f_2 = x_2^2$ $f_3 = \text{sqrt}(2)x_1x_2$

The "Kernel Trick"

$$\max_{\lambda} \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x_i}^{\mathbf{T}} \mathbf{x_j} \right)$$

- The expression for optimization above does not depend on the dimensions of the feature vectors, only their inner ('dot') product.
- We can substitute a kernelized version of the inner product (*k*) for the inner product of feature vectors, where *k* uses a non-linear feature mapping phi: $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$
- After training, we classify according to:

$$g(x) = \sum_{i=1}^{N_s} \lambda_i \, y_i \, k(x_i, x) + w_0$$

R·I·T

Some Common Kernel Functions

Polynomial (d is degree of polynomial)

$$k(x, x') = (x^T x')^d$$

Gaussian

$$k(x, x') = \exp(-||\mathbf{x} - \mathbf{x}'||^2 / 2\sigma^2)$$

Handling Multiple Classes

One vs. All

Create one binary classifier per class

- Most widely used: C (# class) SVMs needed

One vs. One

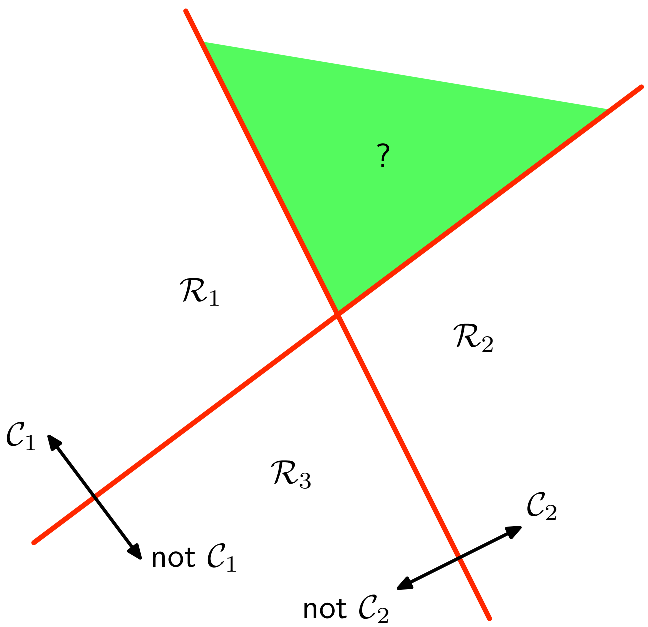
Create one binary classifier for every pair of classes: choose class with highest number of 'votes'

- Variation: use error-correcting output codes (bit strings representing class outcomes), use hamming distance to closest training instances to choose class
- Expensive! (C(C-1)/2 SVMs needed)

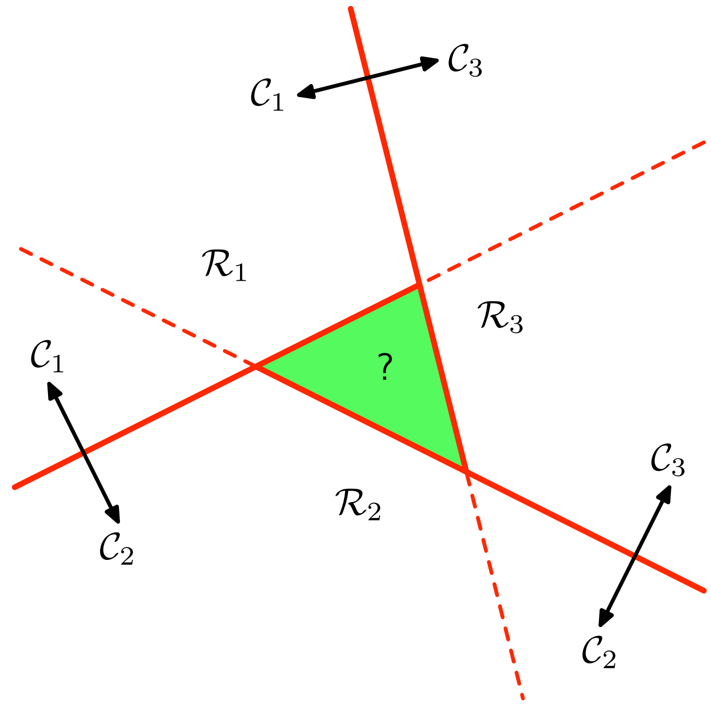
DAGSVM

Organize pair-wise classifiers into a DAG, reduce comparisons

Ambiguous Regions for Combinations of Binary Classifiers



'one vs. all'



'one vs. one'

*Taken from Bishop, "Pattern Recognition and Machine Learning", p. 183