A hyperplane in 2 dimensions:

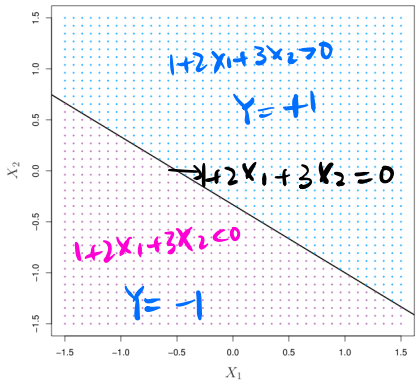
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

A hyperplane in p dimensions:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

define $f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$

separating hyperplane



A separating hyperplane has the property:

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} > 0 \text{ if } y_i = 1,$$

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} < 0 \text{ if } y_i = -1.$$

Equivalent to

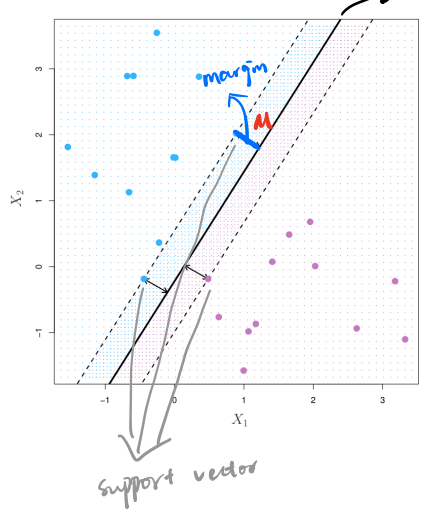
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 0$$

Separate

The Maximal Margin Classifier

Among all separating hyperplanes, find the one that makes the biggest gap or margin between 2 classes

maximal margin hyperplane



We now consider the task of constructing the maximal margin hyperplane based on a set of n training observations $x_1, \ldots, x_n \in \mathbb{R}^p$ and associated class labels $y_1, \ldots, y_n \in \{-1, 1\}$. Briefly, the maximal margin hyperplane is the solution to the optimization problem

$$\underset{\beta_0, \beta_1, \ldots, \beta_p, M}{\text{maximize}} \; M \tag{9.9}$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1, \tag{9.10}$$

(make sure a unique solution for β_j's exits)

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \;\; \forall\, i = 1, \ldots, n. \tag{9.11}$$

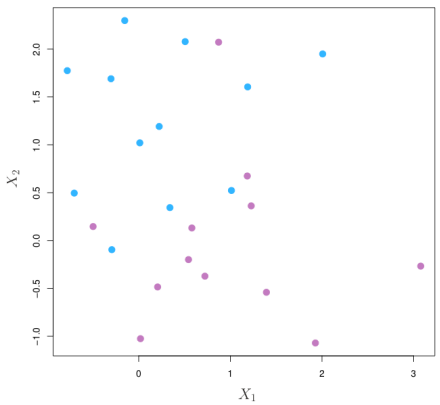
($y_i f(x_i) \geq M$)

(requires each observation on the correct side of hyperplane with some cushion, M is positive)

M represents the margins of hyperplane. Optimazition choose betas to maximize M

The Non-separable Case (cannot use maximum margin classifier)

The Generalization of the maximal margin classifier to the non separable case is support vector classifier.



⇒ 2 classes not separable by a hyperplane. maximum margin classifier cannot be used

Support Vector Classifiers separate with slack

Support vector classifier, a soft margin classifier. The support vector classifier hyperplane is chosen to correctly separate most of the training observations into 2classes, but may misclassify a few observations.

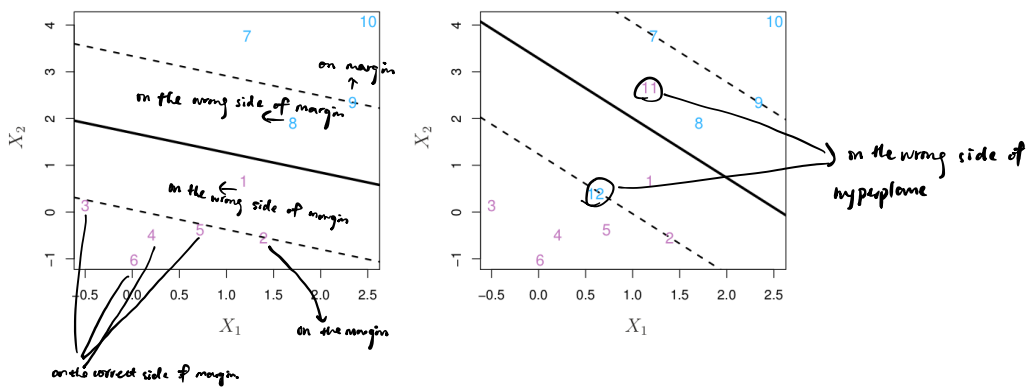
$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M \tag{9.12}$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \tag{9.13}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1-\epsilon_i), \tag{9.14}$$

$$\text{slack variables } \epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C, \tag{9.15}$$

$C \geq 0$. tuning parameter



slack variables $\varepsilon_1 \cdots \varepsilon_n$ allow individual observations to be on the wrong side of margin or hyperplane.

① slack variables ε_i tells where ith observation is located, relative to the hyperplane

$\begin{cases} \varepsilon_i = 0 & \text{ith observation locate on correct side} \\ \varepsilon_i > 0 & \text{... on wrong side, violated margin} \\ \varepsilon_i > 1 & \text{... on wrong hyperplane } (1-\varepsilon_i) < 0 \end{cases}$

② C is budget for the amount that margin can be violated by n observations

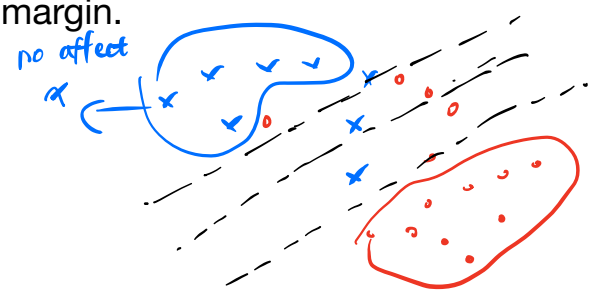
C↑ more tolerant of violations to margin. margin widen

C↓ less .... margin narrow

$\begin{cases} C = 0 & \text{no budget} \quad \varepsilon_1 = \varepsilon_2 = \cdots = \varepsilon_n = 0 \\ C > 0 & \text{no more than } C \text{ observations can be on wrong side of hyperplane} \quad \sum \varepsilon_i \le C \end{cases}$

$\varepsilon_i > 1$ (wrong)

Only observations lie on the margin or violate margin will affect hyperplane. They are support vectors, affect support vector classifier. They don't affect support vector classifier when at correct side of margin.

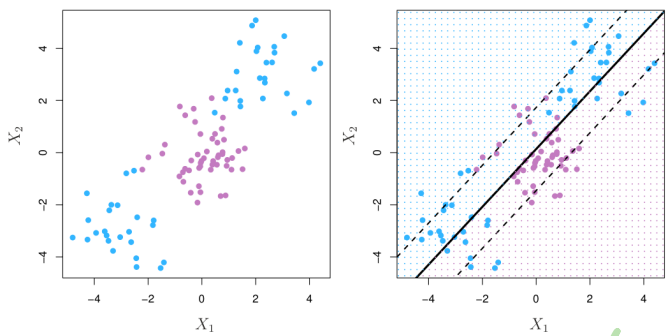


support vector classifiers decision rule based only on a potentially small subset of training observations (support vectors).

Robust to obs far away from hyperplane

Distinct from LDA (depend mean of all obs in each class & in class covariance matrix using all obs)

Logistic Regression also has low sensitivity observations far from decision boundary.



⇒ linear boundary can fail!

nonlinear case

Support Vector Machines

enlarge feature space (eg X_1^2, X_2^2, X_1X_2, $X_1X_2^2$ ···)

p-dim → M dim ($M > p$)

$$\underset{\beta_0,\beta_{11},\beta_{12},\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \; M \tag{9.16}$$

$$\text{subject to } y_i\left(\beta_0 + \sum_{j=1}^{p}\beta_{j1}x_{ij} + \sum_{j=1}^{p}\beta_{j2}x_{ij}^2\right) \ge M(1-\epsilon_i),$$

$$\sum_{i=1}^{n}\epsilon_i \le C, \;\; \epsilon_i \ge 0, \;\; \sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2 = 1.$$

Example: Suppose use $(X_1, X_2, X_1^2, X_2^2, X_1X_2)$ instead (X_1, X_2)

decision boundary: $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 = 0$

nonlinear, solve problem!

Polynomials get wild rather fast ⇒ find elegant controlled way

use of kernels! controlled nonlinear. Kernel!

$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$ (inner product between vectors)

$\left\langle \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} \right\rangle = 1\times0 + 2\times1 + 3\times(-1) = -1$

- The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle, \quad (9.18)$$

test point, training obs ⇒ $\sum_{i=1}^{n} \alpha_i = 0$

where there are n parameters α_i, $i = 1, \ldots, n$, one per training observation.

Intercept: $\beta_0 = \frac{1}{y_j} - \sum_{i=1}^{N} \alpha_i \langle x_i, x_j \rangle$ (y_j: label; x_j: a support vector)

① to evaluate $f(x)$, need to compute inner product between new point x and each of training point

② to estimate $\alpha_1 \cdots \alpha_n$ and β_0 we need $\binom{n}{2}$ inner products $\langle x_i, x_{i'} \rangle$

③ $\alpha_i \neq 0$ only for support vector in solution.

⇒ if a training obs is not a support vector then its $\alpha_i = 0$

$$f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i \langle x, x_i \rangle \quad \sum_{i \in S} \hat{\alpha}_i = 0$$

(support)

$$\beta_0 = \frac{1}{y_j} - \sum_{i \in S} \hat{\alpha}_i \langle x, x_i \rangle$$

S is support set.

Inner product & support vectors:

expand feature space $u = \varphi(x)$ $\dim(u) \neq \dim(x)$

$$f(x) = f(\varphi(x)) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i \langle \varphi(x), \varphi(x_i) \rangle$$

u, u_i; $k(x, x_i)$ kernel

kernel

We don't need to know $\varphi(x)$ to use the kernel.

eg. $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix} = \varphi(x)$

Kernels and Support Vector Machines:

A kernel is a function that quantifies the similarity of two observations.

For SVC, the Kernel is the usual inner product, which is called a *linear kernel*, because $\varphi(x) = x$.

Aside: Mercer's Condition

$k(x_i, x_j)$ is kernel function

⇒ could be written in form $k(x_i, x_j) = [\varphi(x_i)]^T [\varphi(x_j)]$

if and only if it satisfies Mercer's condition

Mercer's condition: for all finite sequences ($x_1, x_2, \ldots, x_n$) and all choices from the sequence of real numbers ($c_1, c_2, \ldots, c_n$) :

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0$$

For example

polynomial kernel

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$$

computes the inner-products needed for d dimensional polynomials-basis functions!

Try it for p = 2 and d = 2.

- The solution has the form

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i).$$

Example:

$x_1 = \begin{bmatrix} u \\ v \end{bmatrix}$ $x_2 = \begin{bmatrix} y \\ z \end{bmatrix}$ ← p=2

$k(x_1, x_2) = (1 + uy + vz)^2 \rightarrow d=2$

$= (1+uy)^2 + (vz)^2 + 2(1+uy)(vz)$

$= 1 + u^2y^2 + 2uy + v^2z^2 + 2vz + 2uyvz$

$f(x) = \beta_0 + \sum_{i=1}^{n} \hat{\alpha}_i k(x, x_i)$

→ $\alpha_i = 0$ if training obs not a support vector

Radial Kernel

$\|x_i - x_{i'}\|^2$

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

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i)$$

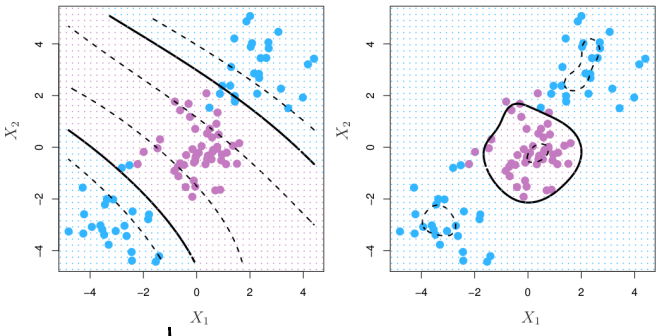
Radial Basis Function (RBF)

Kernel Gaussian

Example: $\left\| \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} - \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} \right\|_2^2 = \left\| \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} \right\| = 3$

Implicit feature space; very high dimensional.

Controls variance by squashing down most dimensions severely.



SVM with polynomial kernel $d = 3$

SVM with radial kernel

if a given test observation $x^* = (x_1^*, \ldots, x_p^*)^T$ is far from training observation x_i in Euclidean distance, then exponent very negative.

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

$$K = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \approx 0$$

γ determines how local the behavior of the RBF kernel is: $\gamma \uparrow \equiv$ small K, $\gamma \downarrow \equiv$ large K

γ allows more training observations to play a role in $f(x^*)$

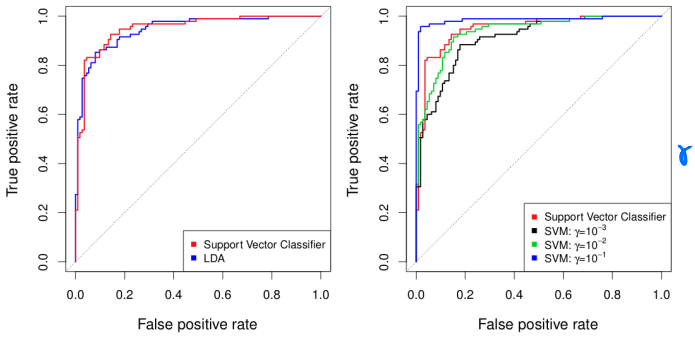
We determine d in the polynomial kernel and γ in RBF. using cross validation

CV (d, C) or (γ, C)

- Therefore, x_i will play virtually no role in $f(x^*)$.
- Recall that the predicted class label for the test observation x^* is based on the sign of $f(x^*)$.
- The radial kernel has very local behavior: only nearby training observations have an effect on the class label of a test observation. (Similar to?) KNN

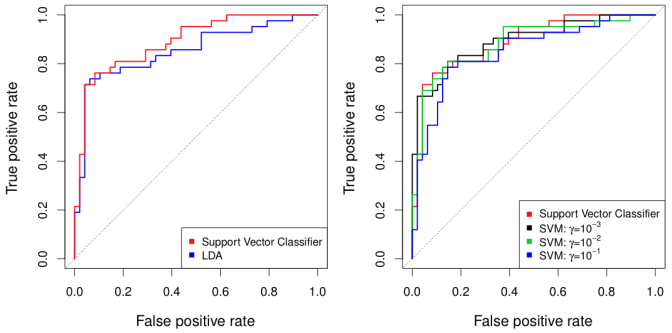
Advantage of using kernel rather than enlarge features:

① computing benefit



ROC for training set.

$\gamma = 10^{-1}$ ☺



$\gamma = 10^{-4}$ ☹

Multiclass classification — SVM for multiple classify

classification task with >2 classes eg. image classify to horse, bird or fish

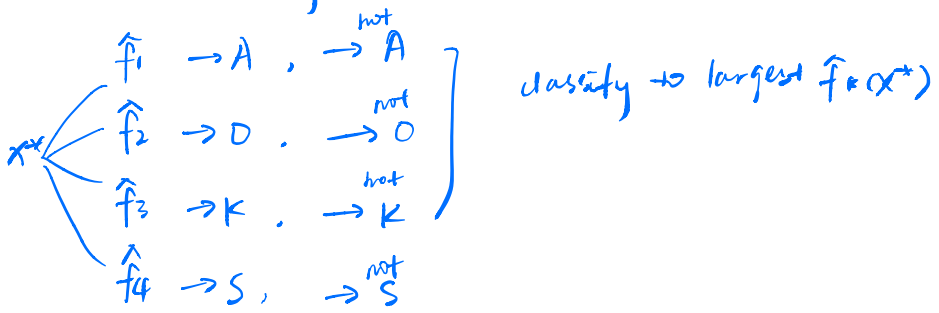
SVMs with More than Two Classes

(OVA) One versus All (The rest)

fit K different 2-class SVM classifier $\hat{f}_k(x)$ $k = 1 \cdots K$

each class versus rest. classify x^* to class for which $\hat{f}_k(x^*)$ is largest.

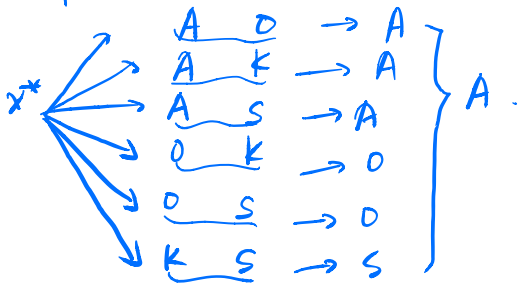
Example: classify Apple. Oranges. kiwis. strawberries



(OVO) one versus one.

fit all $\binom{K}{2}$ pairwise classifier $\hat{f}_{kl}(x)$. classify x^* to class that wins most pairwise. when K is small. choose (OVO)

Example. A D K S 4 class. $\binom{4}{2} = 6$.



Multilabel classification

assigns to each sample a set of target labels. (not only one y)

Example:

A text about religon, politics. finance or education at the same time or none of those.

	religion Y_1	econ Y_2	polit Y_3	Science Y_4
Doc1	0	1	1	0
Doc2	0	0	1	1

- Three methods to solve a multi-label classification problem:
  - Problem Transformation
    - Transform multi-label problem into single-label problem(s)
  - Adapted Algorithms
    - adapting conventional algorithms to directly perform multi-label classification
  - Ensemble approaches
    - Combining multiple classifiers

→ ① Binary relevance ② classifier chains ③ Label Powerset

① Binary relevance

the simplest technique treat each label as separate binary or multi-class classification problem.

X	Y_1	Y_2	Y_3	Y_4
$x^{(1)}$	0	1	1	0
$x^{(2)}$	1	0	0	0
$x^{(3)}$	0	1	0	0
$x^{(4)}$	1	0	0	1

→ Y are target variables

⇒ broken into 4 different single class classification problem

$x^{(5)}$ 0 0 0 1

X	Y_1
$x^{(1)}$	0
$x^{(2)}$	1
$x^{(3)}$	0
$x^{(4)}$	1
$x^{(5)}$	0

X	Y_2
$x^{(1)}$	1
$x^{(2)}$	0
$x^{(3)}$	1
$x^{(4)}$	0
$x^{(5)}$	0

X	Y_3
$x^{(1)}$	1
$x^{(2)}$	0
$x^{(3)}$	0
$x^{(4)}$	0
$x^{(5)}$	0

X	Y_4
$x^{(1)}$	0
$x^{(2)}$	0
$x^{(3)}$	0
$x^{(4)}$	1
$x^{(5)}$	1

all binary class

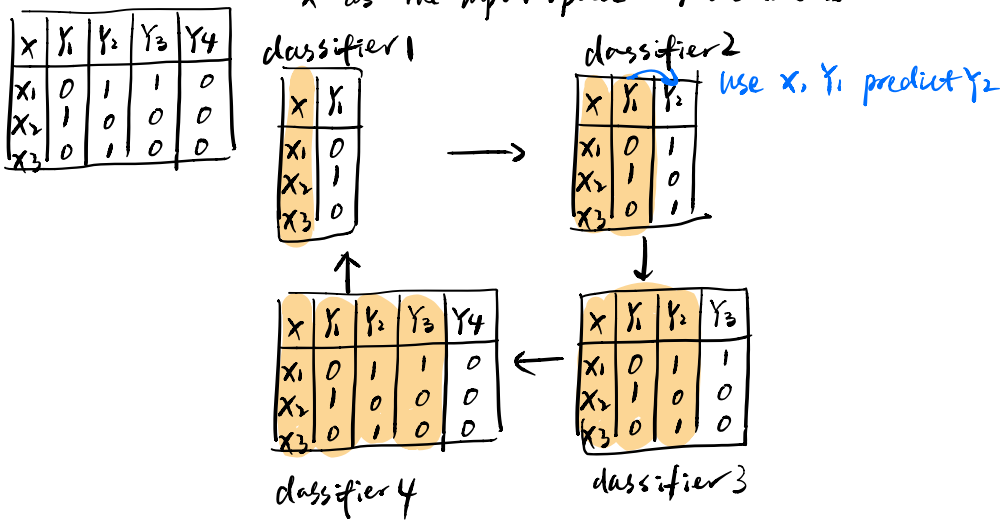
for a new data x^*. each labels $Y_1 \dots Y_4$ is predicted separately.

note1: Any classifier (eg. Naive Bayes Logistic Regression & SVM) can be used for predicting each label

note2: Each label give rise to a binary or multi-class classification problem

② class chains

X as the input space. Y as labels



Similar to binary relevance. the only difference being it forms chains in order to preserve label correlation

⑤ Label Powerset : This method transforms the problem into a multi-class problem with one multi-class classifier is trained on all unique label combinations found in the training data.

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0
x4	0	1	1	0
x5	1	1	1	1
x6	0	1	0	0

→ labels

C_1 x_1 . x_4 has same label

C_2 x_3 . x_6 . . .

C_3

C_1

C_4

C_3

So label powerset transforms a problem into a single multi-class problem

X	Y_1
x_1	1
x_2	2
x_3	3
x_4	1
x_5	4
x_6	3

Drawback:

if too many labels. → will have too many classes some label combination may not in training set or might have only a few samples. so we may encounter class imbalances. or preperation of number of classes

Adapted Algorithms : The most famous adapted algorithm is Multilabel kNN (MLkNN)

label

$l_1 \dots l_n$

- ML-kNN uses the kNN algorithm independently for each label l.
- It finds the k nearest examples to the test instance and considers those that are labeled at least with l as positive and the rest as negative.
- What mainly differentiates this method from other binary relevance (BR) methods is the use of prior probabilities. ML-kNN can also rank labels.
- Decision Trees can be modified to perform multi label classification.
- The main modification is in calculating purities for each region.

Ensemble:

Adaboost MH. AdBoost MR.

Evaluation metrics

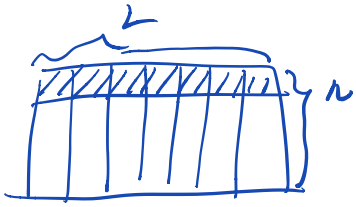
Accuracy score / Exact match metric

This function calculates subset accuracy meaning the predicted set of labels should exactly match with the true set of labels.

Evaluation loss:

Hamming loss : $\frac{1}{NL}\sum_{i=1}^{N}\sum_{j=1}^{L} I(\hat{y}_{ij} \neq y_{ij})$

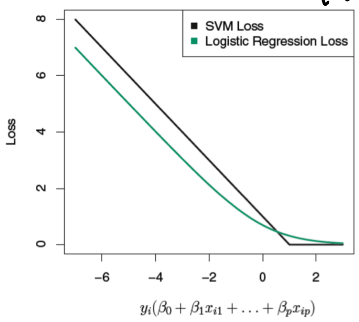
wrong label to total.



With $f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$ one can rephrase support-vector classifier optimization as

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

loss function



This has the form *loss plus penalty*.

The loss is known as the *hinge loss*.

Very similar to "loss" in logistic regression (negative log-likelihood).

$\{\max[0, 1-y_i f(x_i)]\}^2$ → L2 loss.

Squared hinged loss.

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

The above optimization problem has the form of *loss plus penalty*.

Note that the original penalty in SVC is a ridge penalty.

One can change the penalty with L1 penalty and perform variable selection along with Support Vector Classification.

Because the hinge loss function is not differentiable, the solution paths for different values of λ may have jumps.
Therefore, some authors replace the hinge loss with a differentiable squared hinge loss.
Different combinations of hinge loss and squared hinge loss with L1 and L2 penalty result in various versions of SVM.

{ classes very separable. SVM > LR, LDA.
... no ... LR and SVM very similar

SVM popular in high-dimensional. p??n

computation: $O(pn^2)$. for both linear and nonlinear model.

if want to estimate probabilities. LR

or. $p(Y=y_i|X=x_i) \geq \frac{1}{1+e^{y_i f(x_i)}}$

use kernels with LR & Bayesian LDA as well. but computation very expensive.

Kernel try find feature space in which decision boundary are linear. That's not commensurate with characteristics of LR.

(so must use L2 regression with kernel LR)

VC dimension

- The **VC dimension** is a measure of the **capacity** (complexity, expressive power, richness, or flexibility) of a space of functions that can be learned by a classification algorithm.
  A classification model *f* with some parameter vector **β** is said to *shatter* a set of data points $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ if for all assignments of labels to those points, there exists a **β** such that the model *f* makes no errors when evaluating that set of data points.

- The **VC dimension** of a model *f* is the maximum number of points that can be arranged so that *f* shatters them.

VC dimension of straight line for 2p dim.

1 data ✓.

2 data ✓

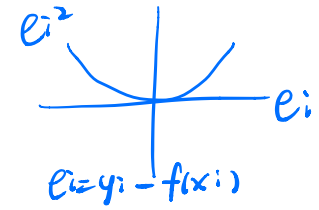
3 points shattered **4 points impossible**

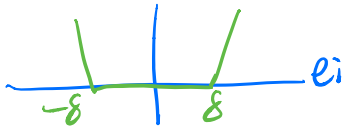
- The VC Dimension of affine classifiers of the form $f(\mathbf{x}) = \boldsymbol{\beta}^T\mathbf{x} + \beta_0$, $\boldsymbol{\beta} \in \mathbb{R}^p$ is $p+1$.
- This includes SVC, a linear SVM.
- The VC Dimension of an SVM equipped with an RBF kernel is infinite.

Support Vector Regression

- There is an extension of the SVM for regression, called support vector regression (SVR) .
- We saw that least squares regression seeks coefficients $\beta_0, \beta_1, \ldots, \beta_p$ such that the sum of squared residuals is as small as possible.
- Support vector regression instead seeks coefficients that minimize a different type of loss, where only residuals larger in absolute value than some positive constant contribute to the loss function.
- This is an extension of the margin used in support vector classifiers to the regression setting.

RSS ↓.





$e_i = y_i - \hat{y}_i$ $= \hat{\beta}_0 + \hat{\beta}_1 x + \cdots$

if error between $-\delta, \delta$ not get account