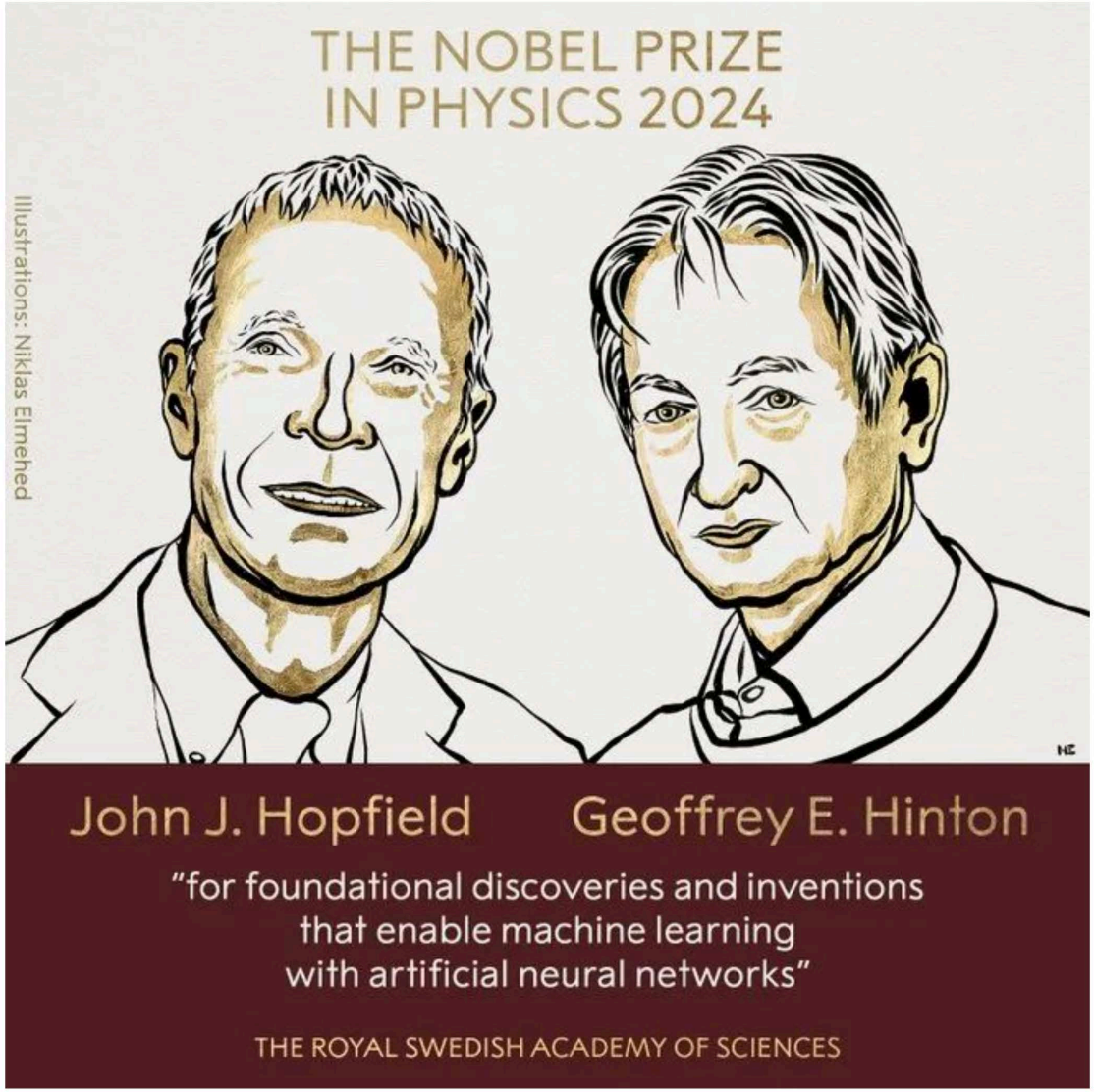
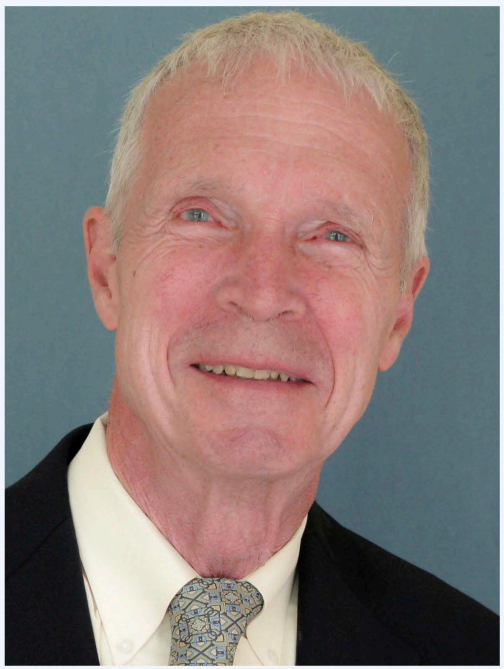
北京时间2024年10月8日17时45分，2024年诺贝尔物理学奖揭晓，获奖者是约翰·霍普菲尔德 (John J. Hopfield) 和杰弗里·辛顿 (Geoffrey E. Hinton)，表彰他们利用人工神经网络在机器学习方面取得的开创性发现和系列发明。





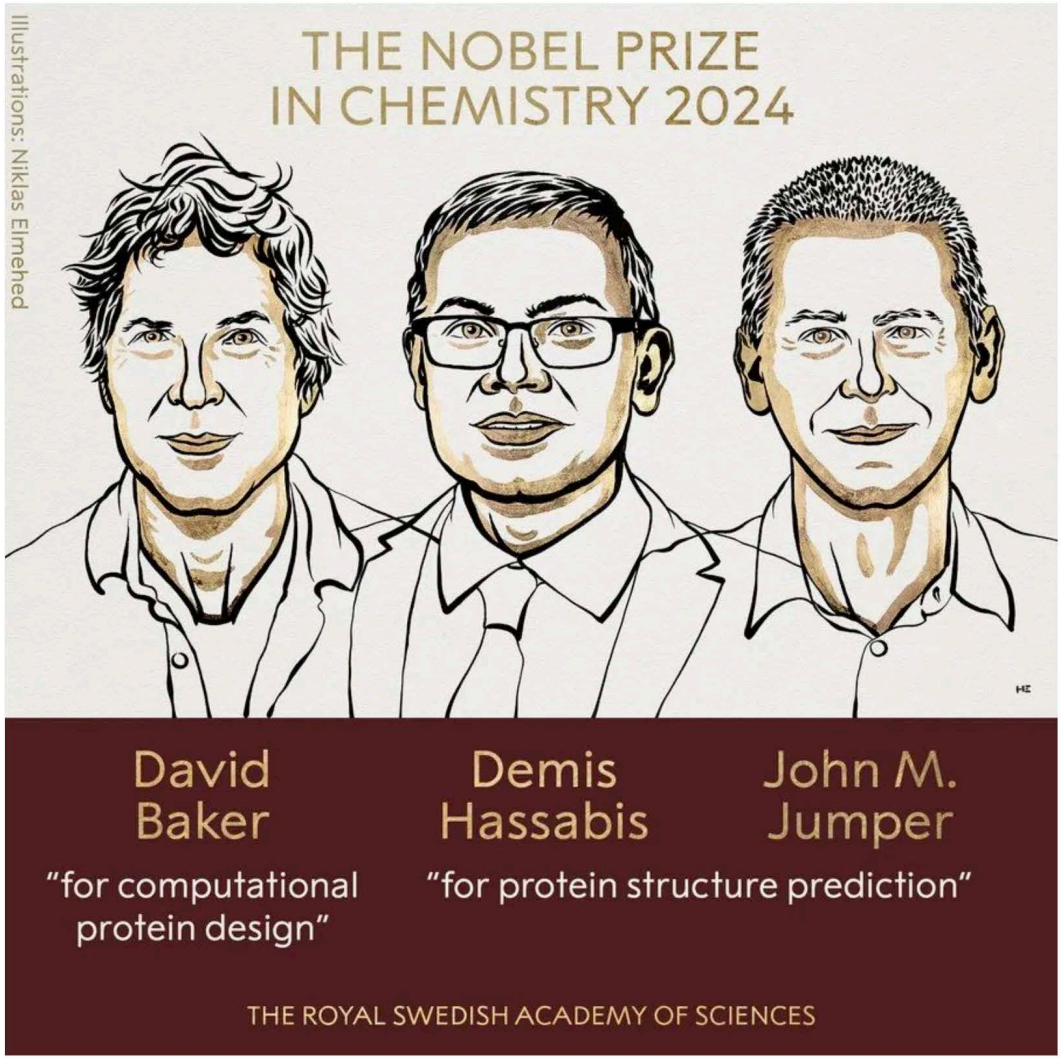
John J. Hopfield，美国生物物理学家，1933年生于美国伊利诺伊州芝加哥，1958年获得康奈尔大学物理学博士学位。现任美国新泽西州普林斯顿大学教授。他在1982年发明了联想神经网络，现在通常称为Hopfield神经网络。Hopfield于1973年当选为美国国家科学院院士，1975年当选为美国艺术与科学学院院士，2022年获得玻尔兹曼奖。

Geoffrey E. Hinton，加拿大计算机科学家和认知心理学家。1947年出生于英国，1978年在爱丁堡大学获得博士学位。曾在谷歌公司工作，现任加拿大多伦多大学教授，多伦多Vector 研究所首席科学顾问。Hinton 于1998年当选为英国皇家学会院士，2016年当选为美国国家工程院外籍院士。2018年，Hinton与Yann LeCun和Yoshua Bengio共同获得了当年的图灵奖。

现在，当人们谈论人工智能时，他们通常指的是使用人工神经网络的机器学习技术。这项技术最初受到了人体大脑结构的启发。在人工神经网络用具有不同值的节点来模仿大脑的神经元。这些节点以类比突触的连接方式相互影响，从而产生更强或更弱的连接。这为训练AI模型提供了最基本的理论之一。今年的物理奖获奖者John Hopfield、Geoffrey Hinton从1980年代开始对人工神经网络进行了重要的先驱研究。

John Hopfield、Geoffrey Hinton开发的技术方法是如今强大的机器学习的先驱。John Hopfield创建了一种联想神经网络，可以存储和重建图像和其他数据模式。而Geoffrey Hinton发明了一种方法，可以自主地发现数据中的特性，从而使“执行识别图像中的特定元素”等任务得以完成。诺贝尔物理学奖委员会主席Ellen Moons称，“他们的工作已经产生了巨大的效益，人们正在相当广泛的领域使用人工神经网络。”

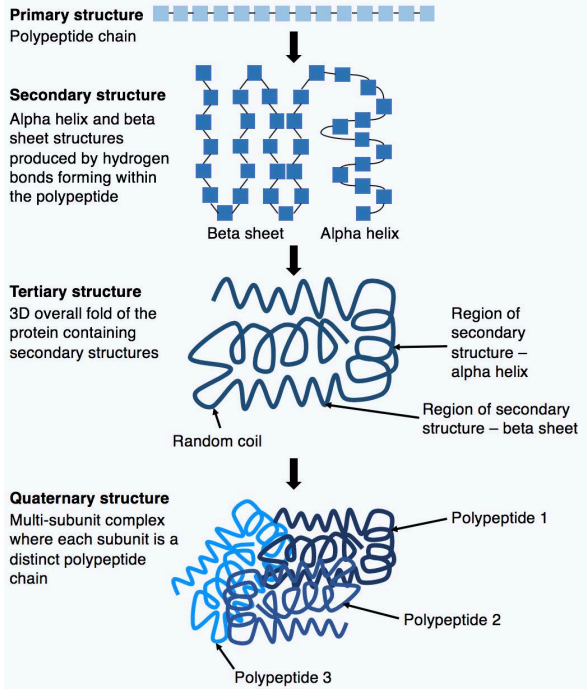
北京时间2024年10月9日17时45分，2024年度诺贝尔化学奖揭晓，一半授予 David Baker，另一半则共同授予 Demis Hassabis 和 John M. Jumper，他们因构建全新蛋白质结构，开发预测蛋白质结构的人工智能模型而受到表彰。



David Baker 美国生物化学家，美国华盛顿大学西雅图分校教授。1962年出生于美国华盛顿州西雅图。1989年获美国加州大学伯克利分校博士学位。他他开创了设计蛋白质和预测其三维结构的方法，开发了蛋白质设计软件，并利用它来创建分子，以解决医学、技术和可持续性方面的挑战。他在2004年获得纽科姆克利夫兰奖、费曼纳米技术奖，在2021年获得了生命科学突破奖。

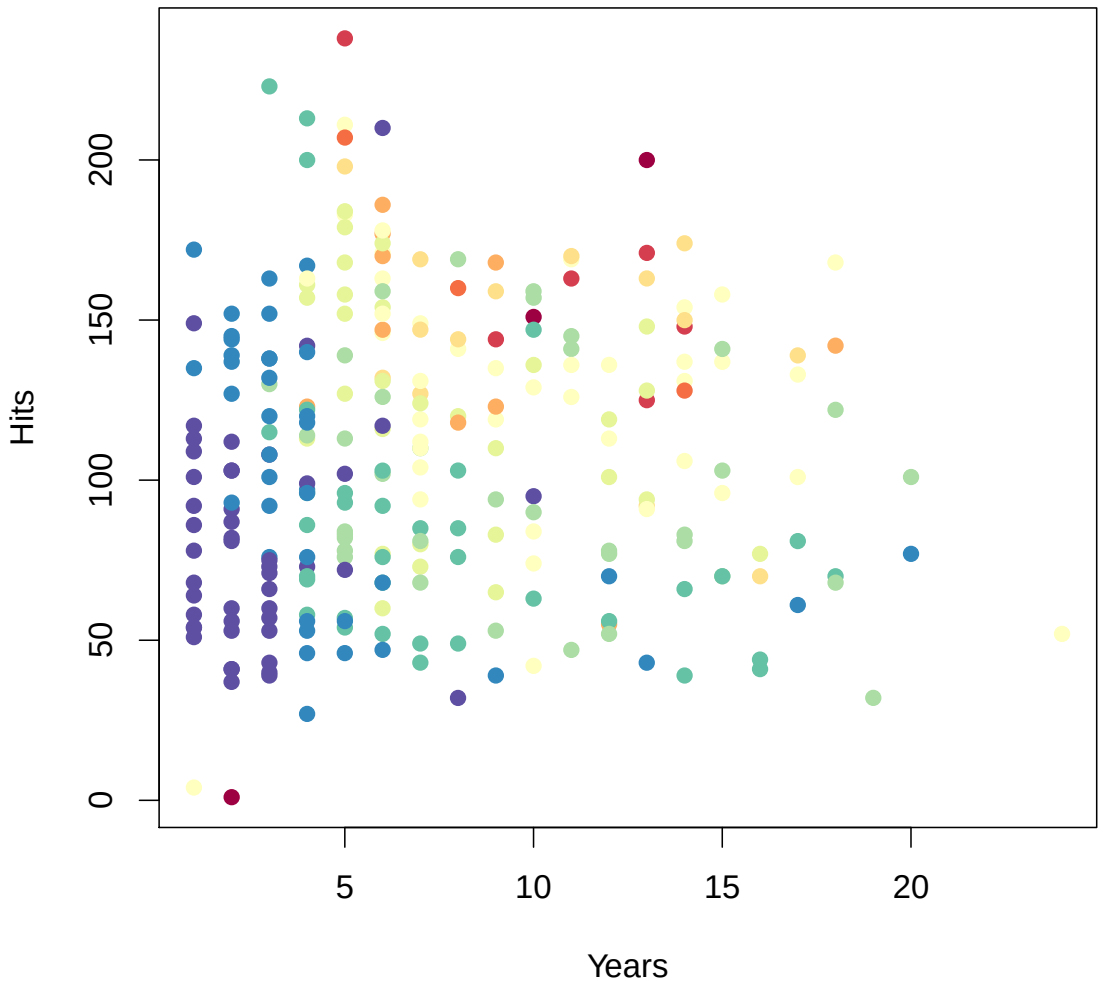
Demis Hassabis，英国计算机学家，1976年生于英国伦敦，毕业于剑桥大学，曾长期从事电子游戏开发。2009年在伦敦大学获得认知神经科学博士学位。2010年与Shane Legg和Mustafa Suleyman一起创立了AI初创公司DeepMind，担任CEO。由该公司开发的人工智能AlphaGo和AlphaFold取得了前所未有的成就，获奖无数，包括科学突破奖、加拿大盖尔德纳国际奖和拉斯克奖。Hassabis 成为英国皇家学会会员。

John M. Jumper，美国计算机学家1985 年出生于美国阿肯色州，2017年在芝加哥大学获得博士学位。自 2018 年起，他在英国伦敦的 Deepmind 担任高级研究科学家。科学杂志《自然》将 John Jumper 列为 2021 年《自然》十大“科学界重要人物”之一。2022年，他获得了威利生物医学科学奖，2023年获得生命科学突破奖。



# Baseball salary data: how would you stratify it?

Salary is color-coded from low (blue, green) to high (yellow,red)



# More details of the tree-building process

- In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or *boxes*, for simplicity and for ease of interpretation of the resulting predictive model.
- The goal is to find boxes $R_1, \ldots, R_J$ that minimize the RSS, given by

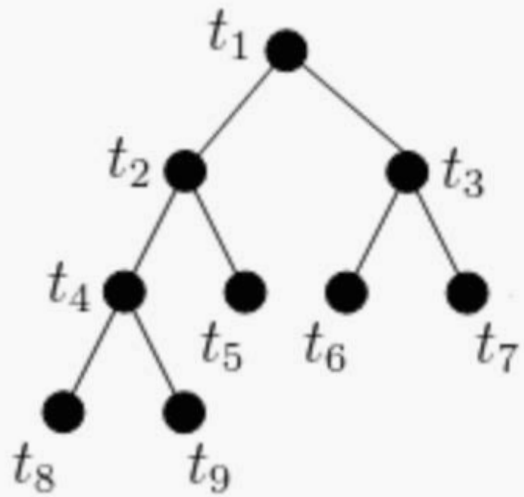
$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the $j$th box.

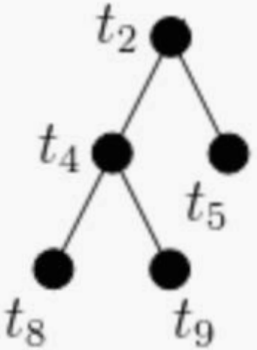
# 上节课回顾

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

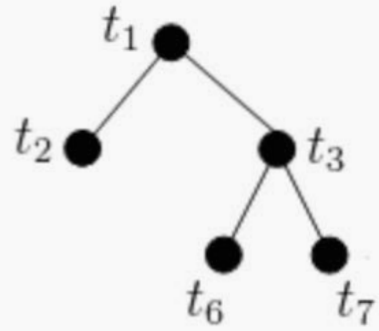
tree $T$




subtree $T_{t_2}$




subtree $T - T_{t_2}$




- Initialization:
  - let $T^1$ be the tree obtained with $\alpha^1 = 0$
  - by minimizing $R(T)$

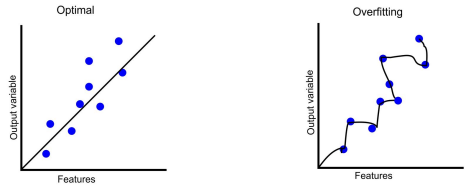
- Step 1
  - select node $t \in T^1$ that minimizes
    - $g_1(t) = \dfrac{R(t) - R(T_t^1)}{|f(T_t^1)| - 1}$
  - let $t_1$ be this node
  - let $\alpha^2 = g_1(t_1)$ and $T^2 = T^1 - T_{t_1}^1$

- step i
  - select node $t \in T^i$ that minimizes
    - $g_i(t) = \dfrac{R(t) - R(T_t^i)}{|f(T_t^i)| - 1}$
  - let $t_i$ be this node
  - let $\alpha^{i+1} = g_i(t_i)$ and $T^{i+1} = T^i - T_{t_i}^i$

Output:

- a sequence of trees $T^1 \supseteq T^2 \supseteq \ldots \supseteq T^k \supseteq \ldots \supseteq \{root\}$
- a sequence of parameters $\alpha^1 \leqslant \alpha^2 \leqslant \ldots \leqslant \alpha^k \leqslant \ldots$

# Pruning a tree




- The process described above may produce good predictions on the training set, but is likely to *overfit* the data, leading to poor test set performance. *Why?*
- A smaller tree with fewer splits (that is, fewer regions $R_1, \ldots, R_J$) might lead to lower variance and better interpretation at the cost of a little bias.
- One possible alternative to the process described above is to grow the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold.
- This strategy will result in smaller trees, but is too *short-sighted:* a seemingly worthless split early on in the tree might be followed by a very good split — that is, a split that leads to a large reduction in RSS later on.

# 上节课回顾

## Summary of Tree Ensemble Methods

- In *bagging*, the trees are grown independently on random samples of the observations. Consequently, the trees tend to be quite similar to each other. Thus, bagging can get caught in local optima and can fail to thoroughly explore the model space.

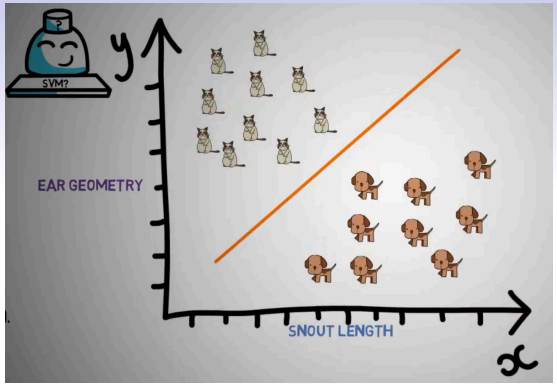
- In *random forests*, the trees are once again grown independently on random samples of the observations. However, each split on each tree is performed using a random subset of the features, thereby decorrelating the trees, and leading to a more thorough exploration of model space relative to bagging.

- In *boosting*, we only use the original data, and do not draw any random samples. The trees are grown successively, using a "slow" learning approach: each new tree is fit to the signal that is left over from the earlier trees, and shrunken down before it is used.

- In *BART*, we once again only make use of the original data, and we grow the trees successively. However, each tree is perturbed in order to avoid local minima and achieve a more thorough exploration of the model space.

观看“支持向量机”介绍视频
https://www.youtube.com/watch?v=Y6RRHw9uN9o

# Support Vector Machines



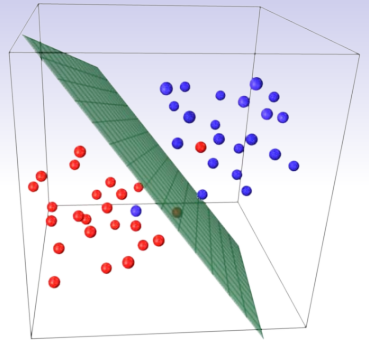

Here we approach the two-class classification problem in a direct way:

> *We try and find a plane that separates the classes in feature space.*

If we cannot, we get creative in two ways:

- We soften what we mean by "separates", and
- We enrich and enlarge the feature space so that separation is possible.

# What is a Hyperplane?



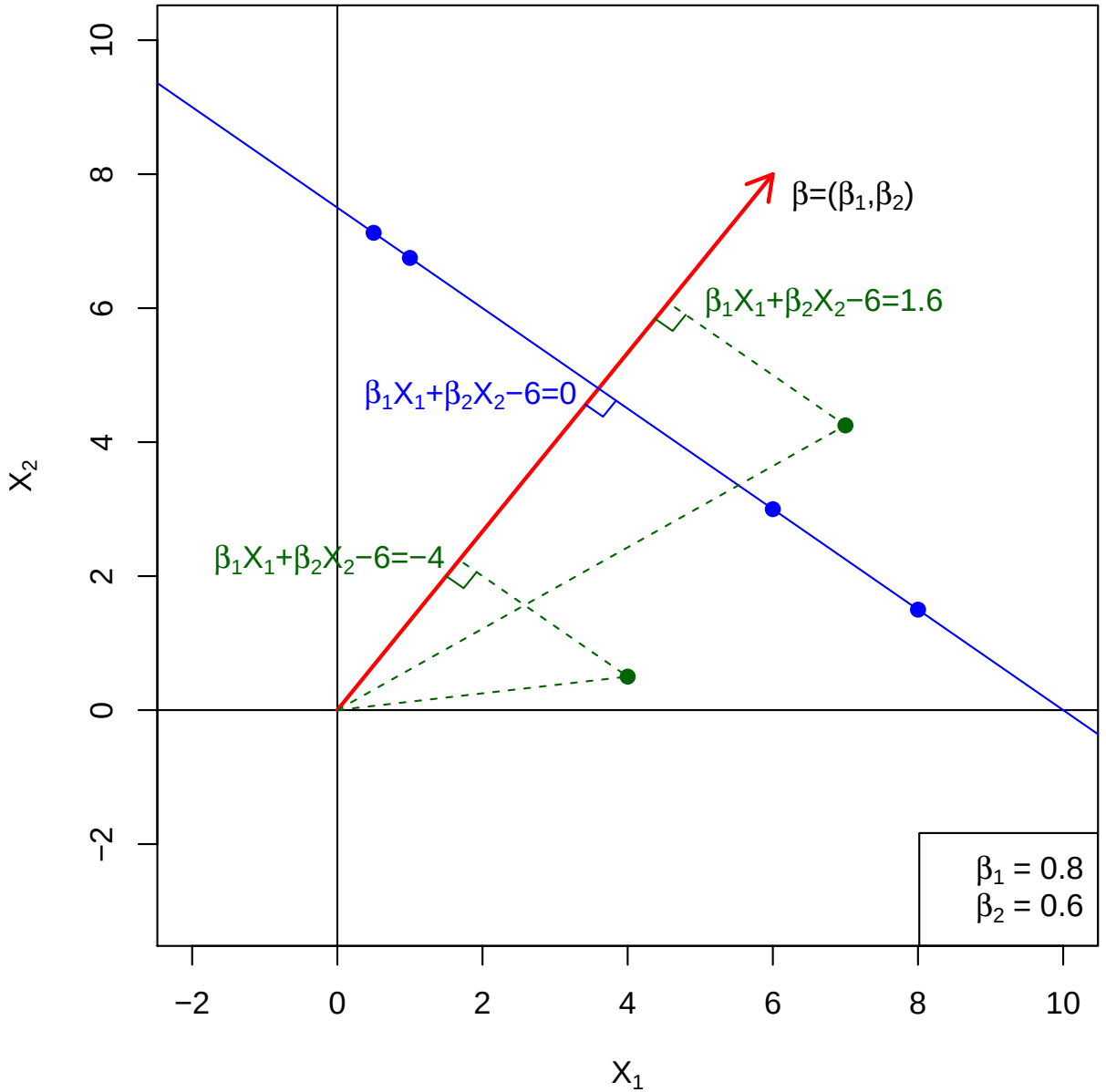
- A hyperplane in $p$ dimensions is a flat affine subspace of dimension $p - 1$.
- In general the equation for a hyperplane has the form

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

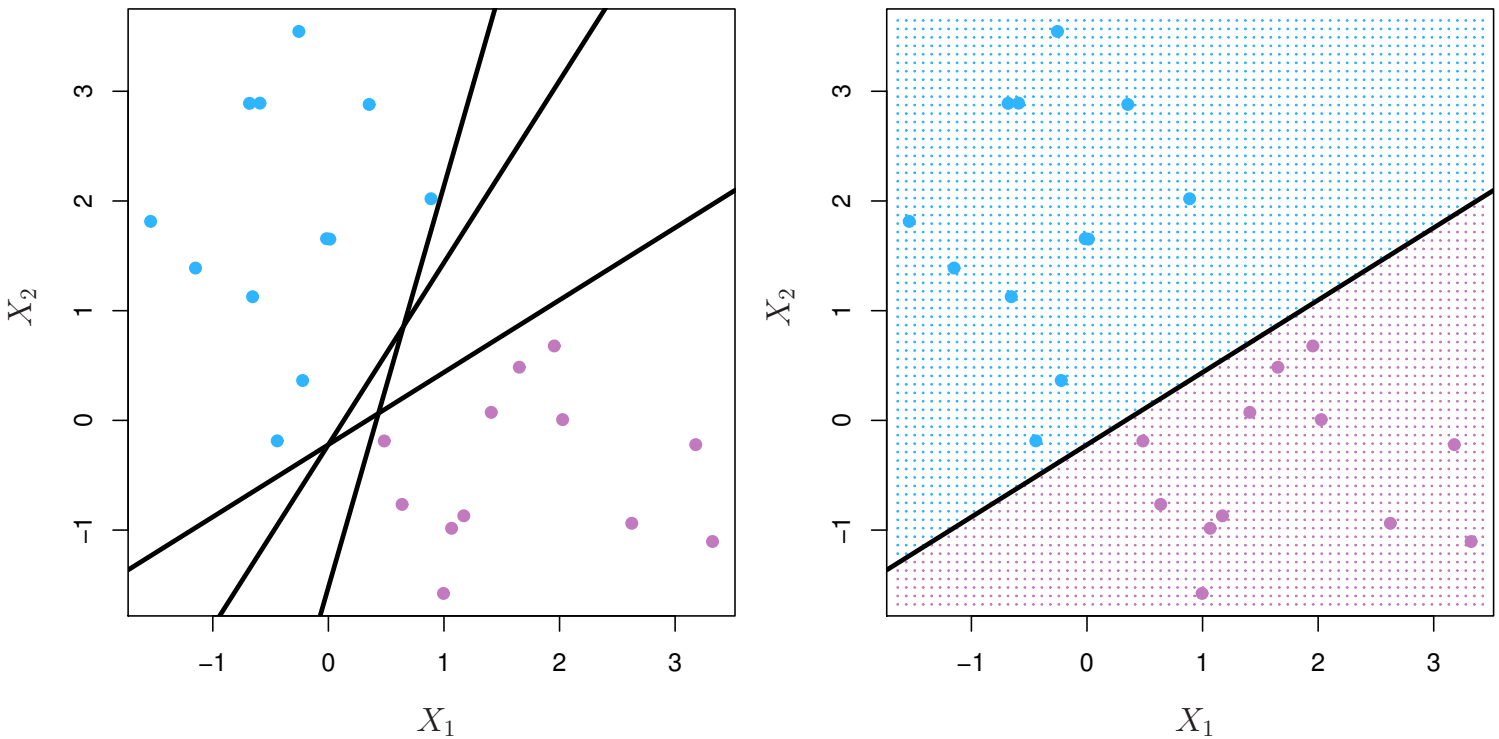
- In $p = 2$ dimensions a hyperplane is a line.
- If $\beta_0 = 0$, the hyperplane goes through the origin, otherwise not.
- The vector $\beta = (\beta_1, \beta_2, \cdots, \beta_p)$ is called the normal vector — it points in a direction orthogonal to the surface of a hyperplane.

what is the distance between two parallel hyperplanes?

# Hyperplane in 2 Dimensions



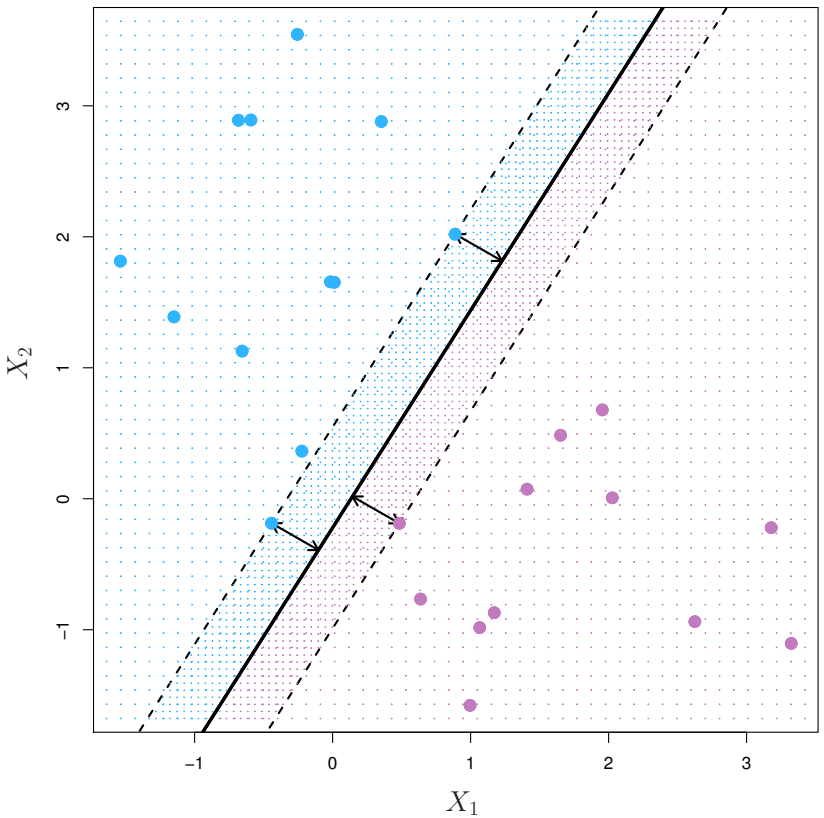
# Separating Hyperplanes




- If $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$, then $f(X) > 0$ for points on one side of the hyperplane, and $f(X) < 0$ for points on the other.
- If we code the colored points as $Y_i = +1$ for blue, say, and $Y_i = -1$ for mauve, then if $Y_i \cdot f(X_i) > 0$ for all $i$, $f(X) = 0$ defines a *separating hyperplane*.

# Maximal Margin Classifier

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




Constrained optimization problem

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

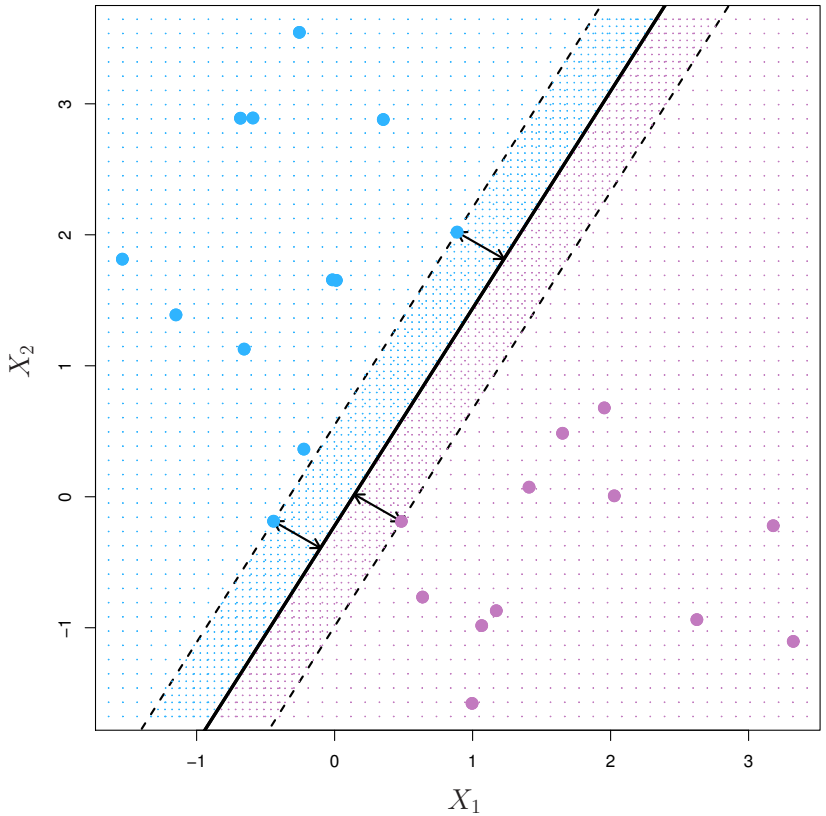
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M$$

$$\text{for all } \; i = 1, \ldots, N.$$

# Maximal Margin Classifier

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




Constrained optimization problem

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

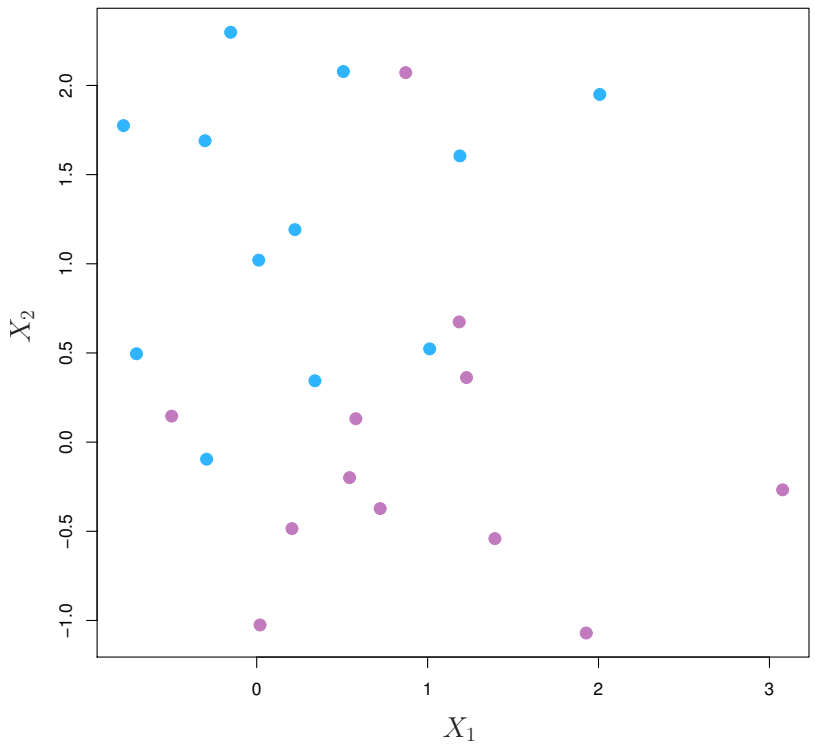
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M$$

$$\text{for all } i = 1, \ldots, N.$$

This can be rephrased as a convex quadratic program, and solved efficiently. The function `svm()` in package `e1071` solves this problem efficiently

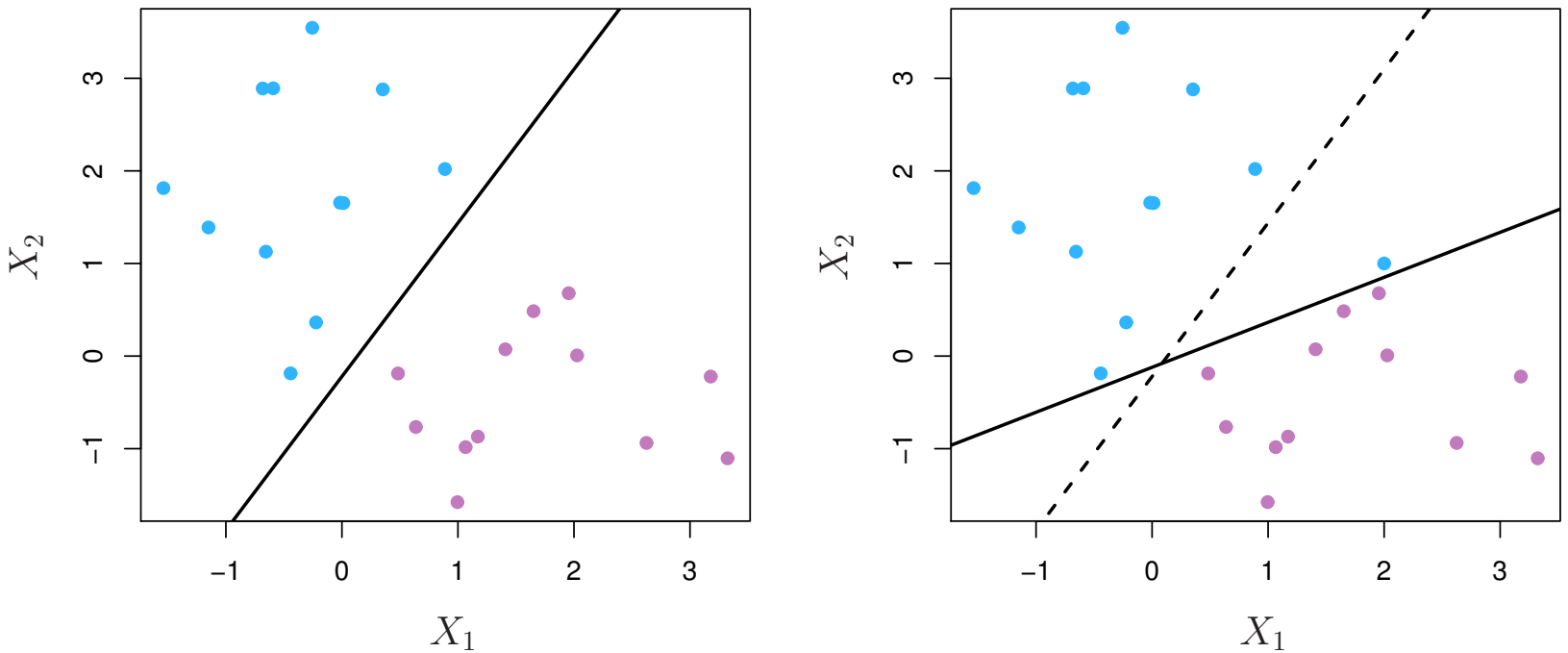
# Non-separable Data




The data on the left are not separable by a linear boundary.

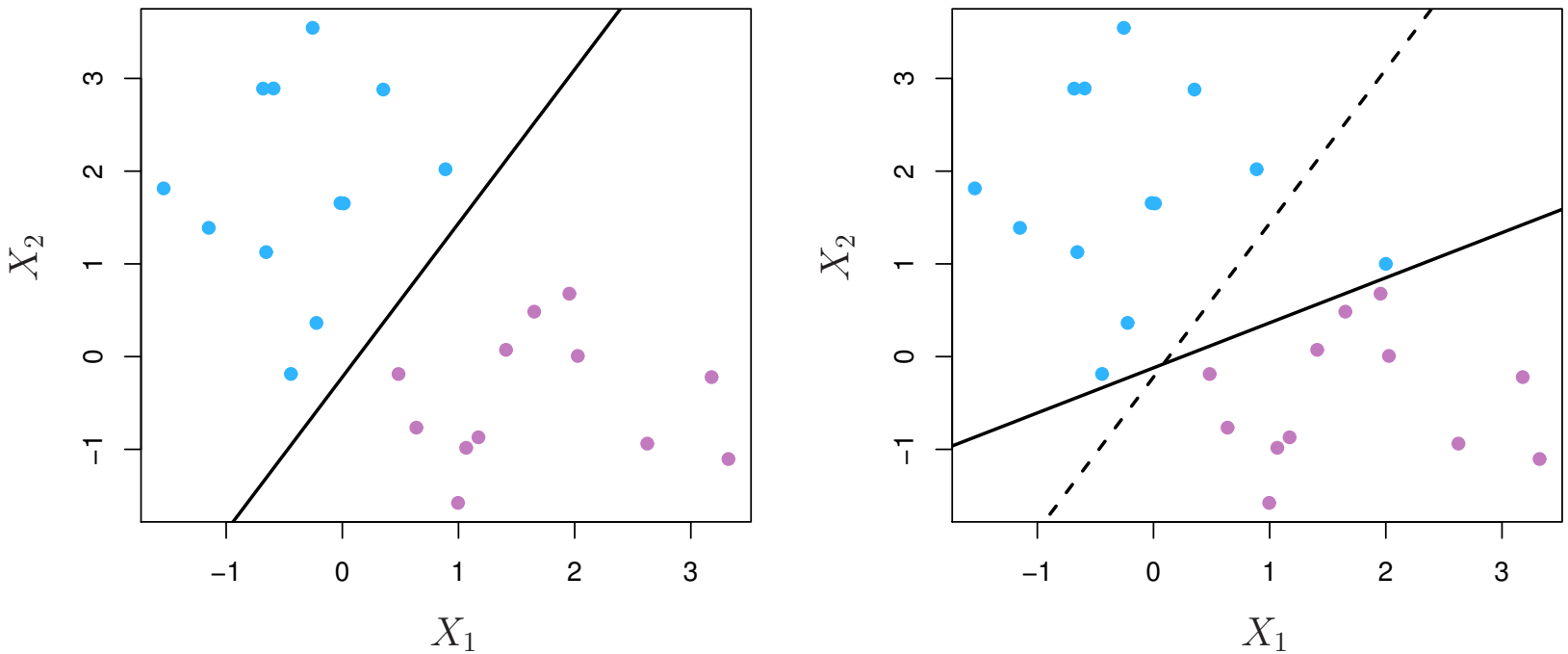
This is often the case, unless $N < p$.

# Noisy Data




Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier.

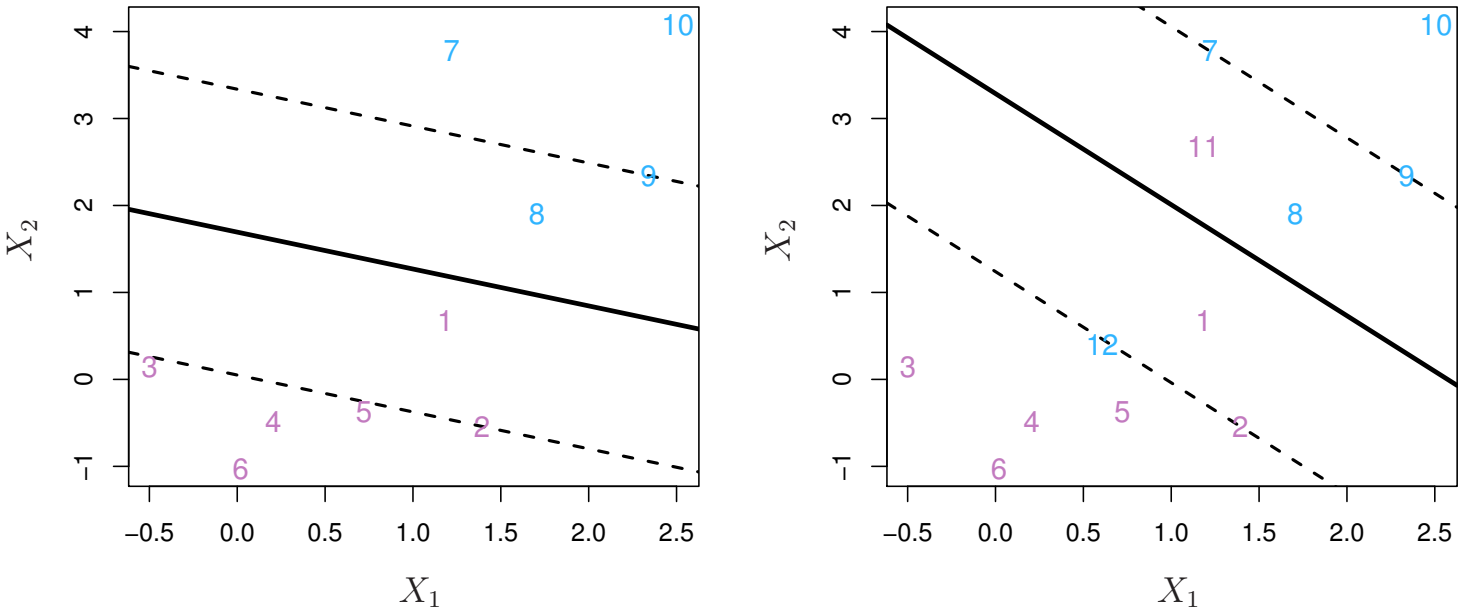
# Noisy Data




Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier.

The *support vector classifier* maximizes a *soft* margin.

# Support Vector Classifier

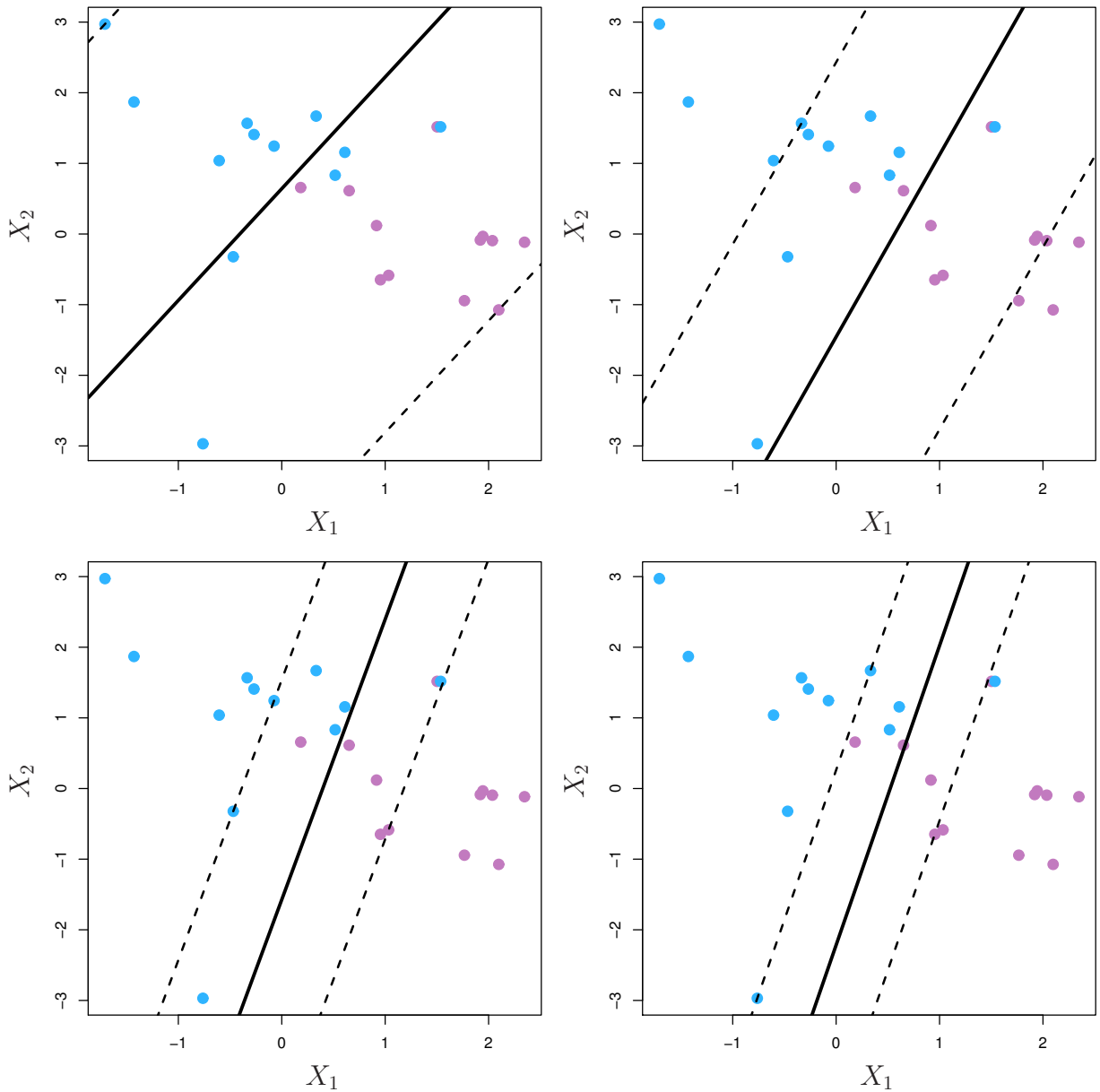



$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \; M \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

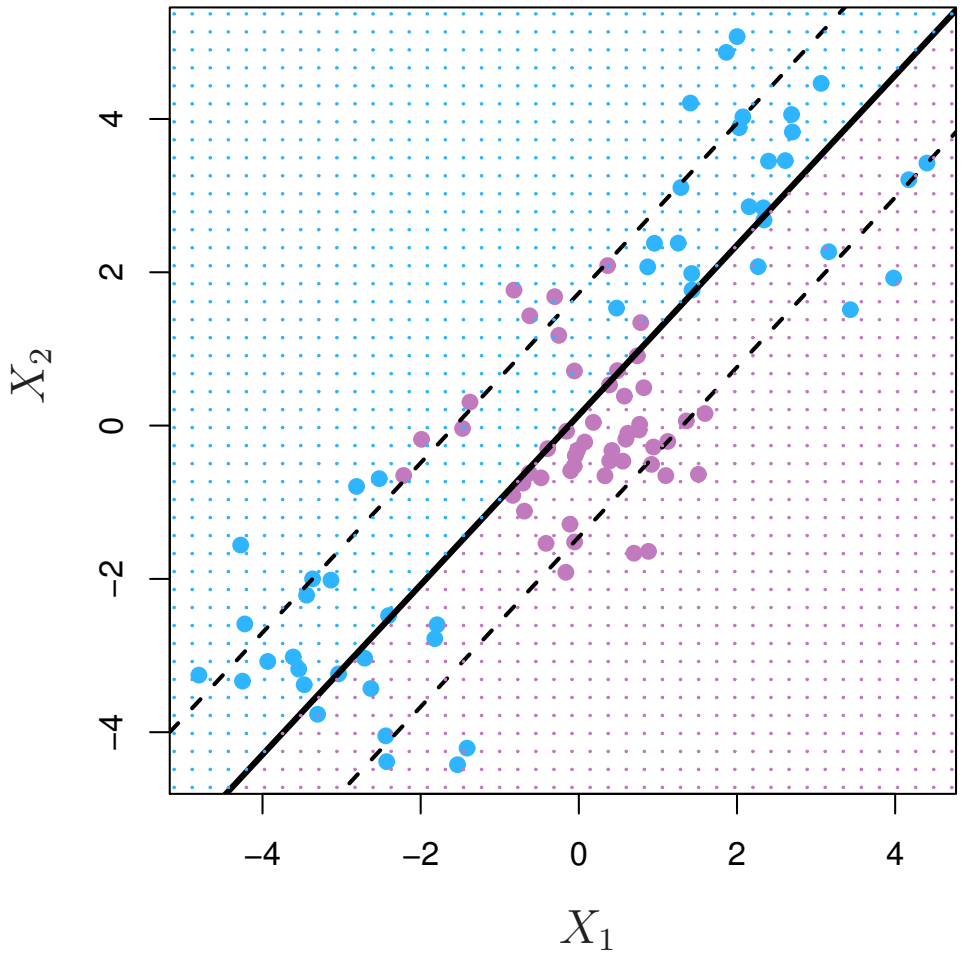
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \; \sum_{i=1}^{n} \epsilon_i \leq C,$$

# $C$ is a regularization parameter



# Linear boundary can fail




Sometime a linear boundary simply won't work, no matter what value of $C$.

The example on the left is such a case.

What to do?

# Feature Expansion

- Enlarge the space of features by including transformations; e.g. $X_1^2$, $X_1^3$, $X_1X_2$, $X_1X_2^2$,.... Hence go from a $p$-dimensional space to a $M > p$ dimensional space.
- Fit a support-vector classifier in the enlarged space.
- This results in non-linear decision boundaries in the original space.

# Feature Expansion

- Enlarge the space of features by including transformations; e.g. $X_1^2$, $X_1^3$, $X_1X_2$, $X_1X_2^2$,.... Hence go from a $p$-dimensional space to a $M > p$ dimensional space.
- Fit a support-vector classifier in the enlarged space.
- This results in non-linear decision boundaries in the original space.

Example: Suppose we use $(X_1,\ X_2,\ X_1^2,\ X_2^2,\ X_1X_2)$ instead of just $(X_1, X_2)$. Then the decision boundary would be of the form

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

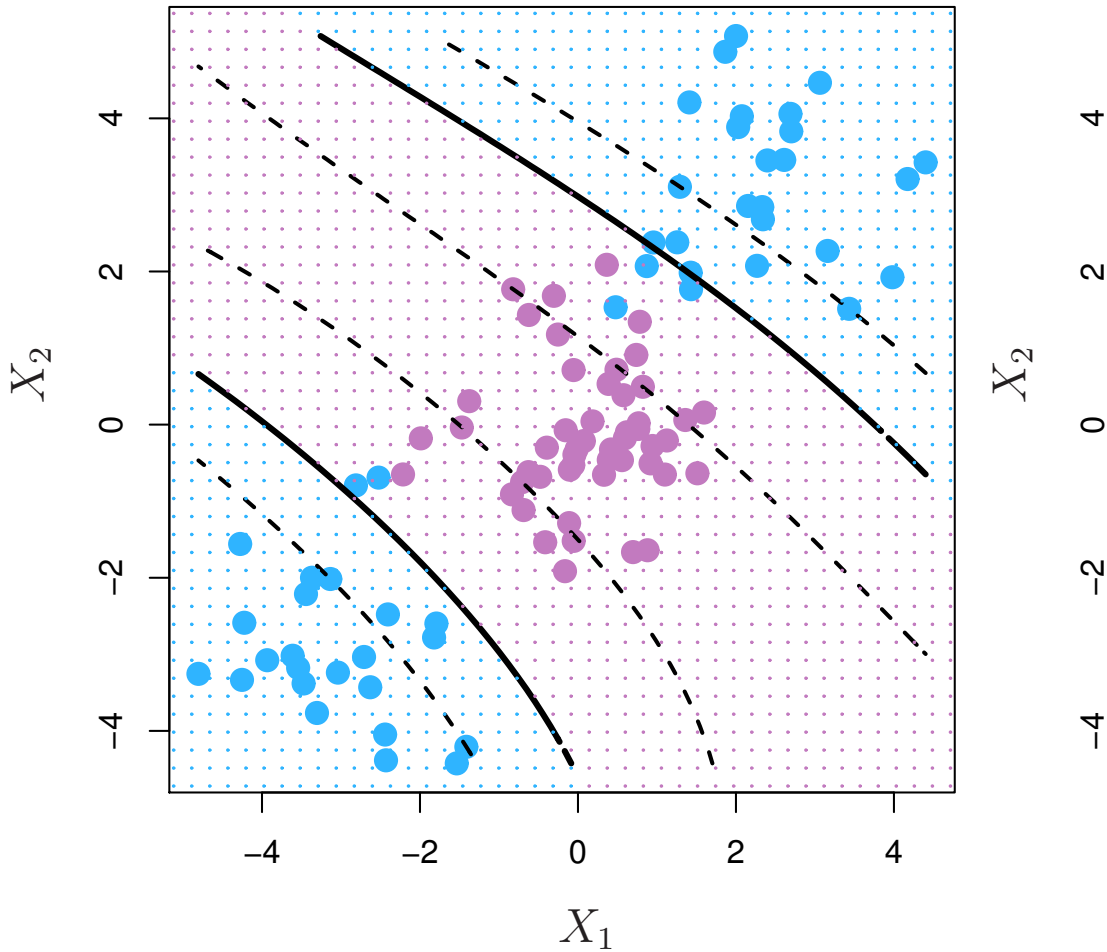
This leads to nonlinear decision boundaries in the original space (quadratic conic sections).

# Cubic Polynomials

Here we use a basis expansion of cubic polynomials

From 2 variables to 9

The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space

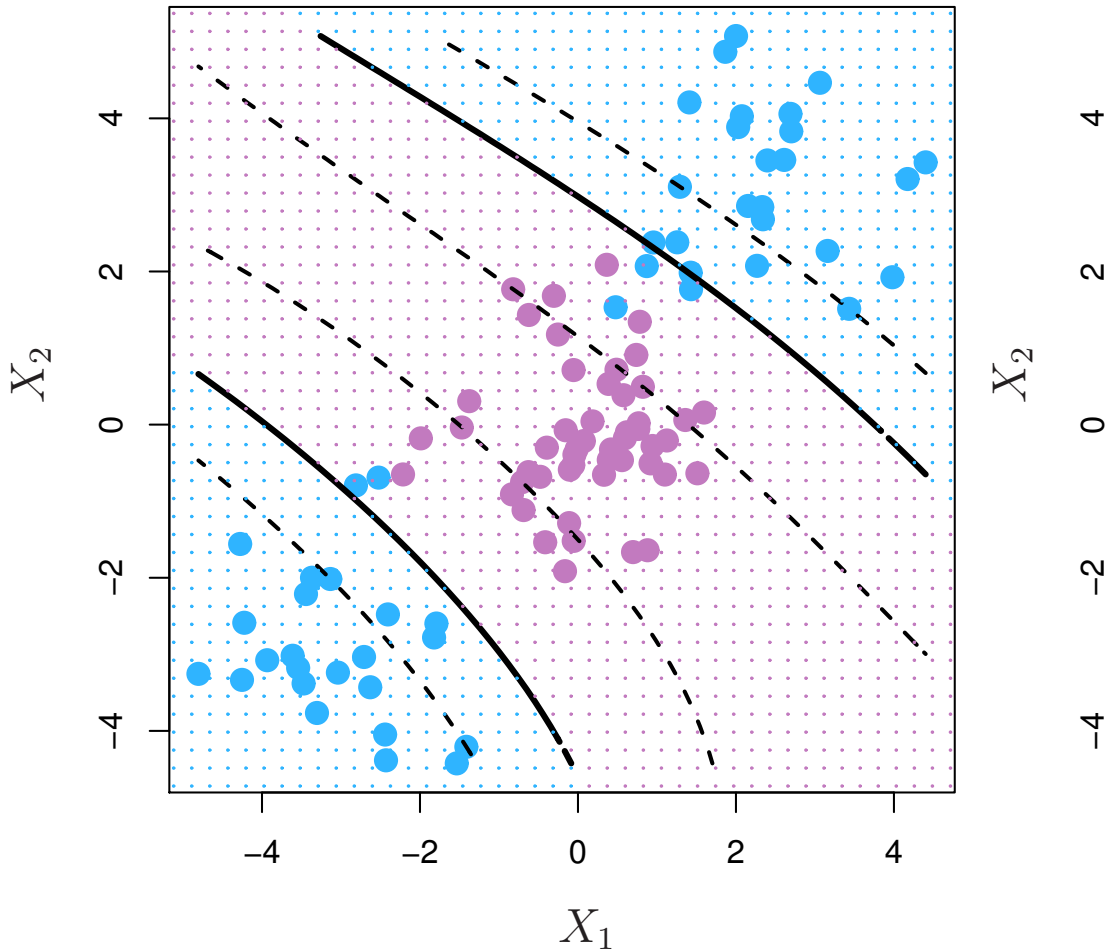


# Cubic Polynomials

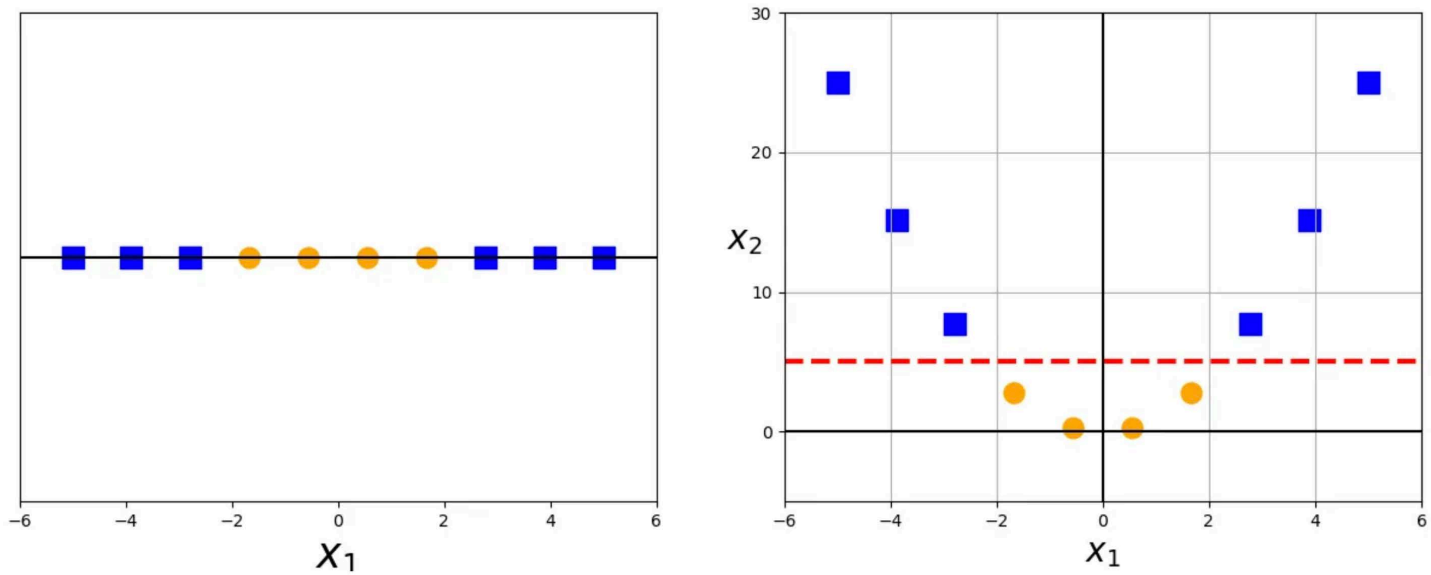
Here we use a basis expansion of cubic polynomials

From 2 variables to 9

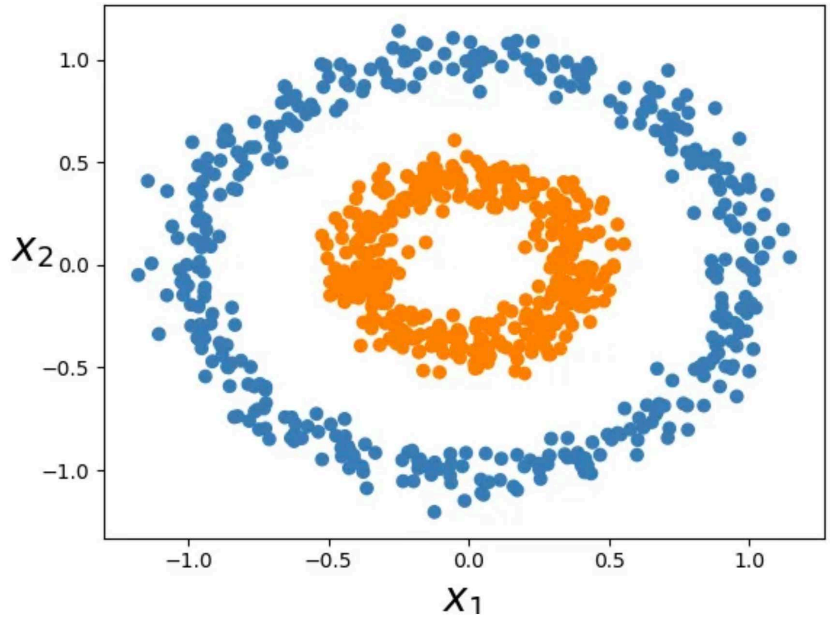
The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space

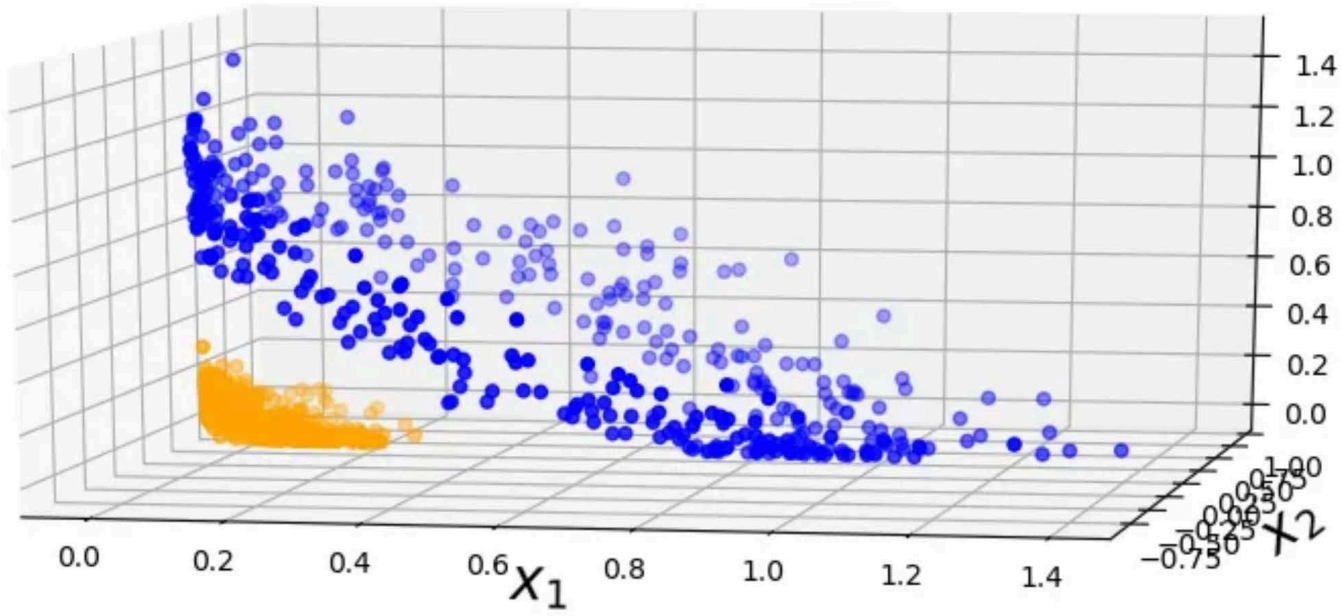



$$\beta_0+\beta_1X_1+\beta_2X_2+\beta_3X_1^2+\beta_4X_2^2+\beta_5X_1X_2+\beta_6X_1^3+\beta_7X_2^3+\beta_8X_1X_2^2+\beta_9X_1^2X_2=0$$




This data becomes linearly separable after a quadratic transformation to 2-dimensions.

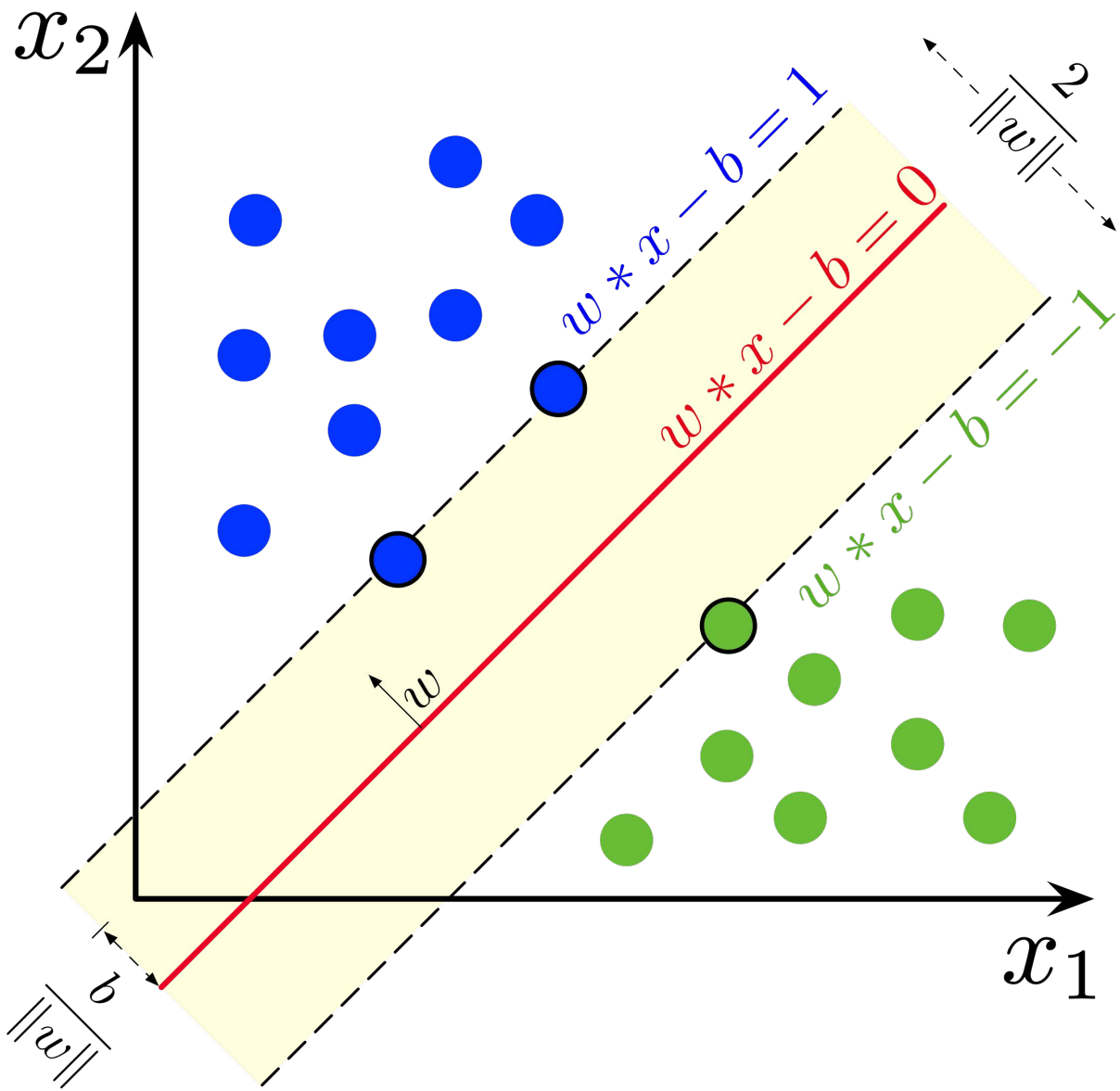

1.0
0.5
0.0
−0.5
−1.0
$x_2$
−1.0
−0.5
0.0
0.5
1.0
$x_1$


0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
$x_1$
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.75
$x_2$

# Nonlinearities and Kernels

- Polynomials (especially high-dimensional ones) get wild rather fast.
- There is a more elegant and controlled way to introduce nonlinearities in support-vector classifiers — through the use of *kernels*.
- Before we discuss these, we must understand the role of *inner products* in support-vector classifiers.

# 寻找hyperplane的优化过程



SVM 要做的就是确保所有正类样本都在 $\mathrm{w}^{\mathrm{T}}\mathrm{x}+\mathrm{b}=1$ 的上方，所有负类样本都在 $\mathrm{w}^{\mathrm{T}}\mathrm{x}+\mathrm{b}=-1$ 的下方的情况下，最大化直线 $\mathrm{w}^{\mathrm{T}}\mathrm{x}+\mathrm{b}+1=0$ 和直线 $\mathrm{w}^{\mathrm{T}}\mathrm{x}+\mathrm{b}-1=0$ 之间的间距。根据几何知识，我们得到这两条直线的距离 $\mathrm{d}=\frac{2}{||\mathrm{w}||}$。我们需要求解 $\mathrm{d}$ 的最大值，等价于求 $\frac{1}{2}||\mathrm{w}||^2$ 的最小值。因此，要优化的目标可写作如下形式，

$$\begin{aligned}&\arg\min_{<\mathrm{w},\mathrm{b}>}\ \frac{1}{2}||\mathrm{w}||^2\\&\mathrm{s.t.}\ \mathrm{y}^{(\mathrm{i})}(\mathrm{w}^{\mathrm{T}}\mathrm{x}^{(\mathrm{i})}+\mathrm{b})\geq 1,\ \forall\,\mathrm{i}\in[1,\mathrm{m}]\end{aligned}$$

SVM 的优化思路是 **样本点到超平面的间隔最大化**，在原空间上SVM 要优化的问题的拉格朗日函数如下所示，即我们要做的就是在 $\lambda_{\mathrm{i}}\geq 0$ 的约束下求解 $\mathrm{L}(\mathrm{w},\mathrm{b},\lambda)$ 函数的最小值， (详细的推导过程请参考我的前一篇博客 → 软硬SVM)

$$\begin{aligned}&\mathrm{L}(\mathrm{w},\mathrm{b},\lambda)=\frac{1}{2}||\mathrm{w}||^2+\sum_{\mathrm{i}}^{\mathrm{m}}\lambda_{\mathrm{i}}\left[1-\mathrm{y}^{(\mathrm{i})}(\mathrm{w}^{\mathrm{T}}\mathrm{x}^{(\mathrm{i})}+\mathrm{b})\right]\\&\mathrm{s.t.}\ \lambda_{\mathrm{i}}\geq 0\end{aligned}$$

# 对偶问题

SVM 的优化思路是 **样本点到超平面的间隔最大化**，在原空间上SVM 要优化的问题的拉格朗日函数如下所示，即我们要做的就是在 $\lambda_i \geq 0$ 的约束下求解 $L(w, b, \lambda)$ 函数的最小值，(详细的推导过程请参考我的前一篇博客 → 软硬SVM)

$$L(w, b, \lambda) = \frac{1}{2}||w||^2 + \sum_i^m \lambda_i [1 - y^{(i)}(w^T x^{(i)} + b)]$$
$$s.t.\ \lambda_i \geq 0$$

我们将其转换为对偶问题 (即先对 $w, b$ 求 $L$ 的最小值，在对 $\lambda$ 求 $L$ 的最大值)，对 $w, b$ 的求导得到，

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^m \lambda_i y^{(i)} \phi(x^{(i)}),\quad \frac{\partial L}{\partial b} = -\sum_{i=1}^m \lambda_i y^{(i)}$$

令 $w, b$ 的偏导数为 0，我们得到 $w^* = \sum_{i=1}^m \lambda_i y^{(i)} \phi(x^{(i)})$ 以及 $\sum_{i=1}^m \lambda_i y^{(i)} = 0$。我们将它们带入到 $L$ 中可得到其最小值 $L_{min}(w, b, \lambda) = \theta(\lambda)$，最优化问题转变为求解 $\theta(\lambda)$ 的最大值问题，即

$$\theta(\lambda) = -\frac{1}{2}\sum_{i=1}^m \sum_{j=1}^m \lambda_i \lambda_j y^{(i)} y^{(j)} [\phi(x_i)]^T \phi(x_j) + \sum_{i=1}^m \lambda_i$$
$$s.t. \sum_{i=1}^m \lambda_i y^{(i)} = 0,\ \lambda_i \geq 0, i = 1...m.$$

SMO序列最小优化算法

> 由于 $\phi(x_i), \phi(x_j)$ 都是 (nx1) 维向量，因此上式中的 $[\phi(x_i)]^T \phi(x_j)$ 也可以写为内积的形式 $\phi(x_i) \cdot \phi(x_j)$，故 $\theta(\lambda)$ 也可以写成 $\theta(\lambda) = -\frac{1}{2}\sum_{i=1}^m \sum_{j=1}^m \lambda_i \lambda_j y^{(i)} y^{(j)} \phi(x_i)\phi(x_j) + \sum_{i=1}^m \lambda_i$。

查看链接https://blog.csdn.net/chikily_yongfeng/article/details/105645955

*Equation 5-9. Kernel trick for a $2^{nd}$-degree polynomial mapping*

$$\phi(\mathbf{a})^T \cdot \phi(\mathbf{b}) = \begin{pmatrix} a_1^2 \\ \sqrt{2}\, a_1 a_2 \\ a_2^2 \end{pmatrix}^T \cdot \begin{pmatrix} b_1^2 \\ \sqrt{2}\, b_1 b_2 \\ b_2^2 \end{pmatrix} = a_1^2 b_1^2 + 2 a_1 b_1 a_2 b_2 + a_2^2 b_2^2$$

$$= (a_1 b_1 + a_2 b_2)^2 = \left( \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^T \cdot \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right)^2 = \left( \mathbf{a}^T \cdot \mathbf{b} \right)^2$$

On the left-hand side, we have the dot product of the transformed feature vectors, which is equal to our 2nd-degree polynomial kernel function.

# Inner products and s

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \quad \text{— } \textit{inner}$$

- Solution of the dual problem gives us:
$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i$$
- The decision boundary:
$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0$$
- The decision:
$$\hat{y} = \text{sign}\left[ \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0 \right]$$
- Mapping to a feature space, we have the decision:
$$\hat{y} = \text{sign}\left[ \sum_{i \in SV} \hat{\alpha}_i y_i (\phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i)) + w_0 \right]$$

- The linear support vector classifier can be represented as

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

- To estimate the parameters $\alpha_1, \ldots, \alpha_n$ and $\beta_0$, all we need are the $\binom{n}{2}$ inner products $\langle x_i, x_{i'} \rangle$ between all pairs of training observations.

It turns out that most of the $\hat{\alpha}_i$ can be zero:

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

$\mathcal{S}$ is the *support set* of indices $i$ such that $\hat{\alpha}_i > 0$. [see slide 8]

# Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!

# Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!
- Some special *kernel functions* can do this for us. E.g.

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

computes the inner-products needed for $d$ dimensional polynomials — $\binom{p+d}{d}$ basis functions!

# Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!
- Some special *kernel functions* can do this for us. E.g.

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

computes the inner-products needed for $d$ dimensional polynomials — $\binom{p+d}{d}$ basis functions!

*Try it for $p = 2$ and $d = 2$.*

# Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!
- Some special *kernel functions* can do this for us. E.g.

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

computes the inner-products needed for $d$ dimensional polynomials — $\binom{p+d}{d}$ basis functions!

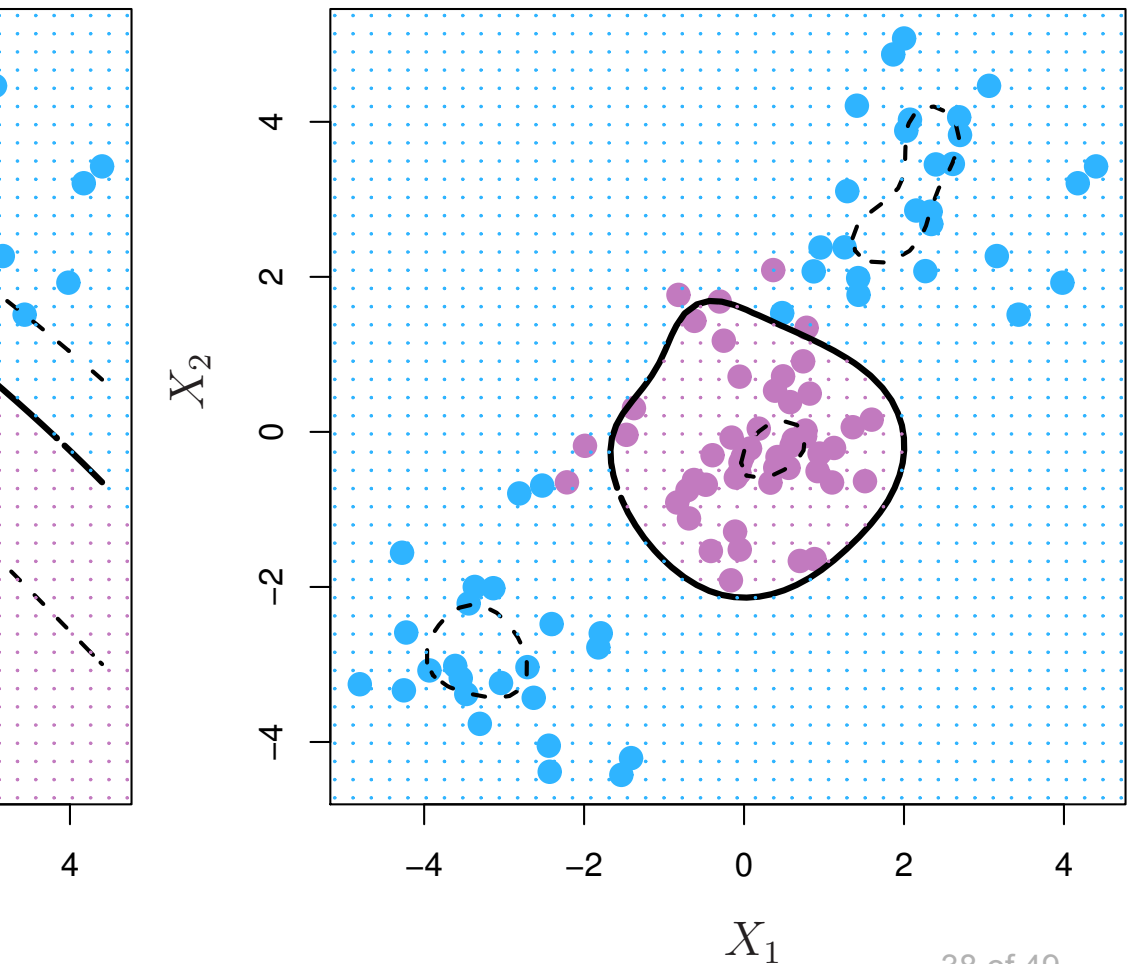
*Try it for $p = 2$ and $d = 2$.*

- The solution has the form

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

# Radial Kernel

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

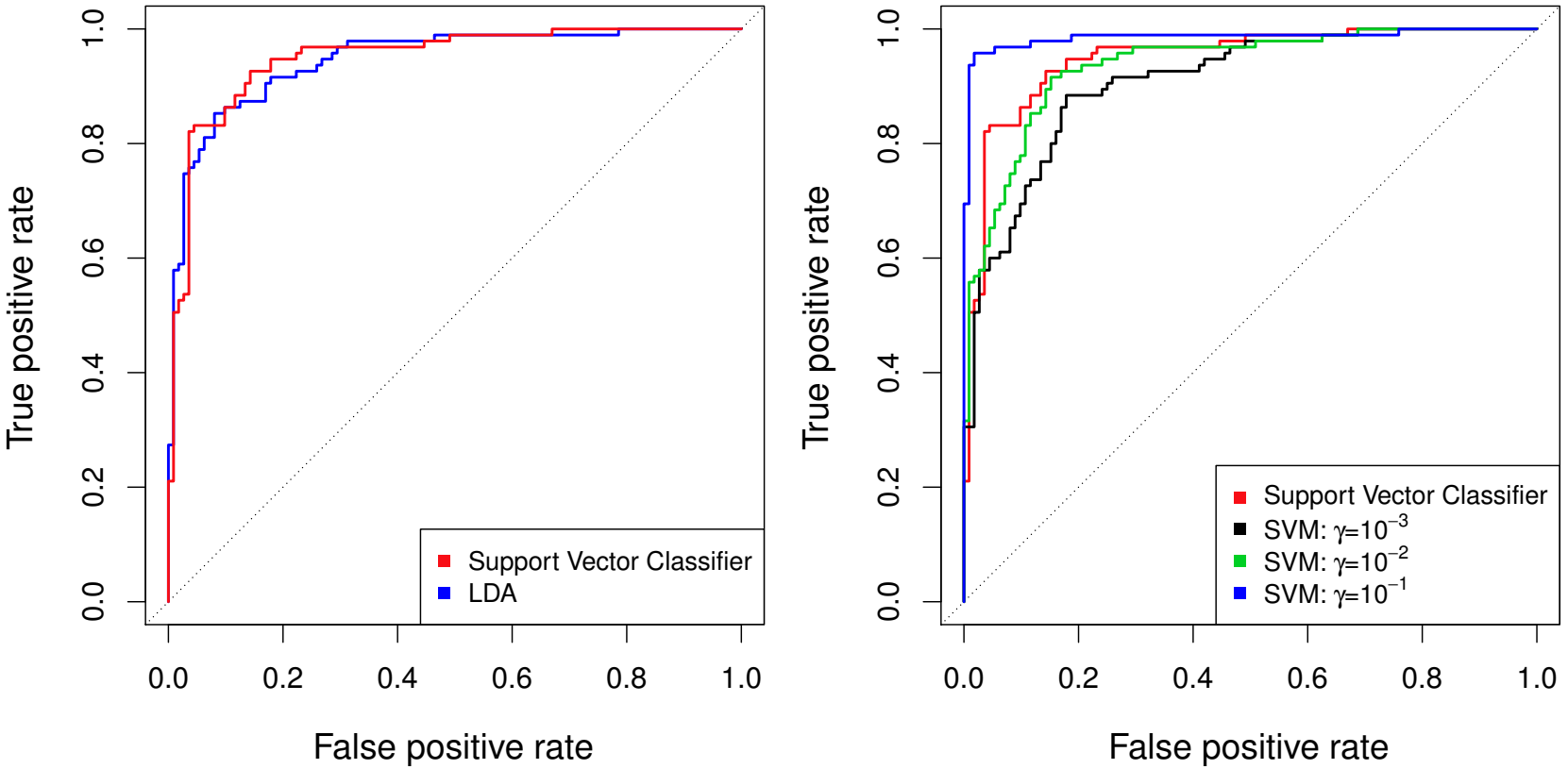



$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i)$$

Implicit feature space; very high dimensional.

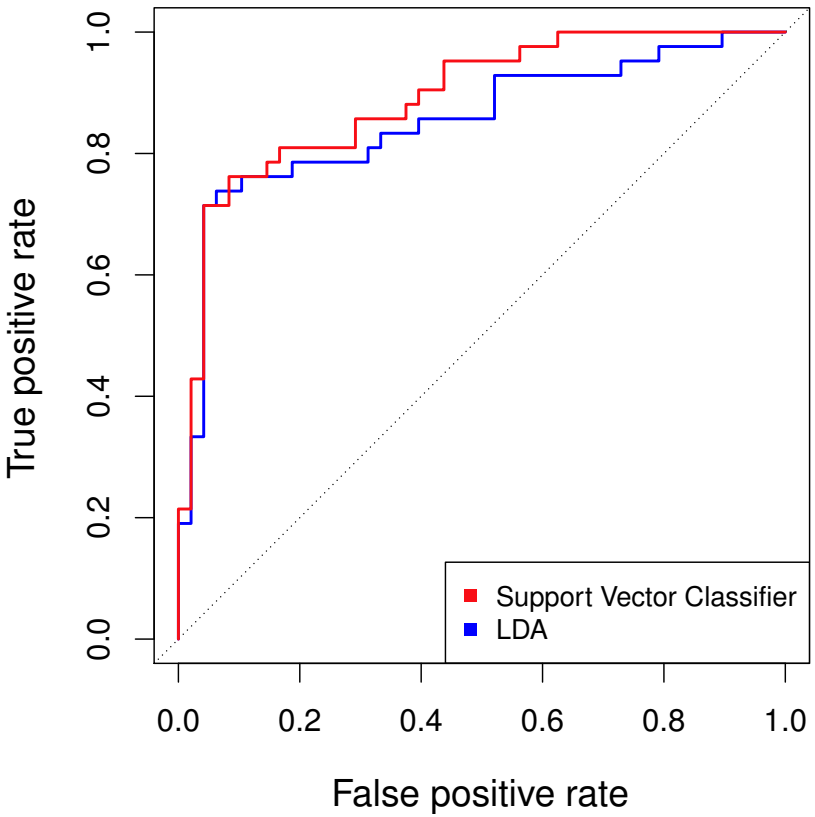
Controls variance by squashing down most dimensions severely

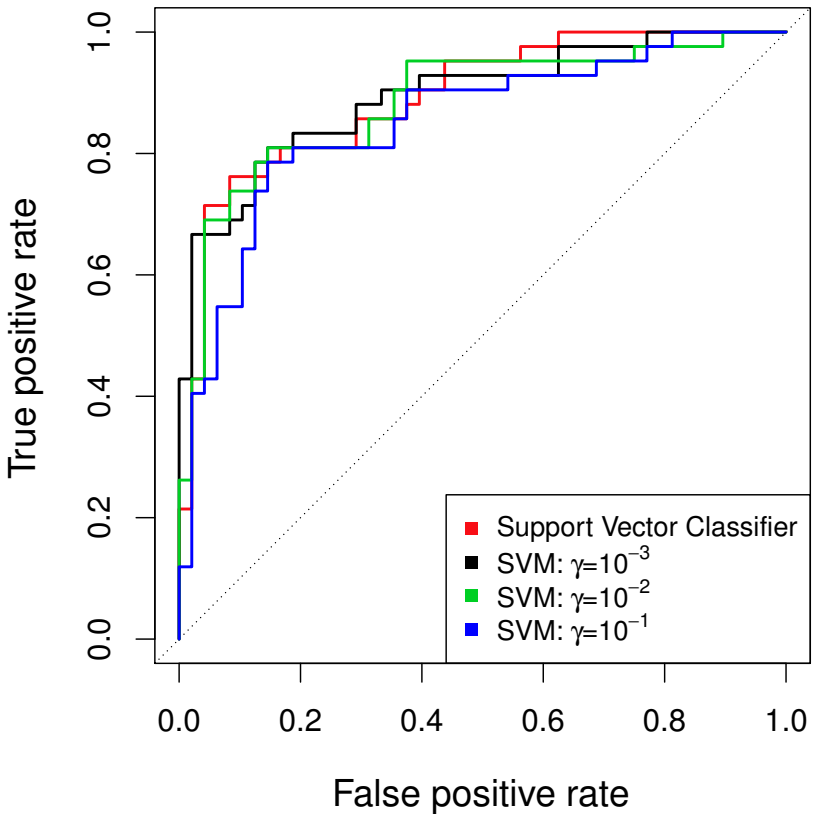
# Example: Heart Data




ROC curve is obtained by changing the threshold 0 to threshold $t$ in $\hat{f}(X) > t$, and recording *false positive* and *true positive* rates as $t$ varies. Here we see ROC curves on training data.

# Example continued: Heart Test Data



# SVMs: more than 2 classes?

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?

# SVMs: more than 2 classes?

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?

**OVA** One versus All. Fit $K$ different 2-class SVM classifiers $\hat{f}_k(x)$, $k = 1, \ldots, K$; each class versus the rest. Classify $x^*$ to the class for which $\hat{f}_k(x^*)$ is largest.

# SVMs: more than 2 classes?

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?

**OVA** One versus All. Fit $K$ different 2-class SVM classifiers $\hat{f}_k(x)$, $k = 1, \ldots, K$; each class versus the rest. Classify $x^*$ to the class for which $\hat{f}_k(x^*)$ is largest.

**OVO** One versus One. Fit all $\binom{K}{2}$ pairwise classifiers $\hat{f}_{k\ell}(x)$. Classify $x^*$ to the class that wins the most pairwise competitions.

# SVMs: more than 2 classes?

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?

**OVA** One versus All. Fit $K$ different 2-class SVM classifiers $\hat{f}_k(x)$, $k = 1, \ldots, K$; each class versus the rest. Classify $x^*$ to the class for which $\hat{f}_k(x^*)$ is largest.

**OVO** One versus One. Fit all $\binom{K}{2}$ pairwise classifiers $\hat{f}_{k\ell}(x)$. Classify $x^*$ to the class that wins the most pairwise competitions.

Which to choose? If $K$ is not too large, use OVO.

Available online at www.sciencedirect.com

ScienceDirect

Advances in Space Research 65 (2020) 1263–1278

ADVANCES IN SPACE RESEARCH (a COSPAR publication)

www.elsevier.com/locate/asr

# Developing a dust storm detection method combining Support Vector Machine and satellite data in typical dust regions of Asia

Lamei Shi [a,b,c], Jiahua Zhang [a,b,c,*], Da Zhang [a,b,c], Tertsea Igbawua [b,d], Yuqin Liu [e]

[a] *Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100094, China*
[b] *Key Laboratory of Digital Earth Science, Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing 100094, China*
[c] *University of Chinese Academy of Sciences, Beijing 101407, China*
[d] *University of Agriculture Makurdi, PMB 2373 Markurdi, Benue State, Nigeria*
[e] *Key Lab of Urban Environment and Health, Institute of Urban Environment, Chinese Academy of Sciences, Xiamen 361021, China*

Received 23 April 2019; received in revised form 24 October 2019; accepted 20 November 2019
Available online 28 November 2019

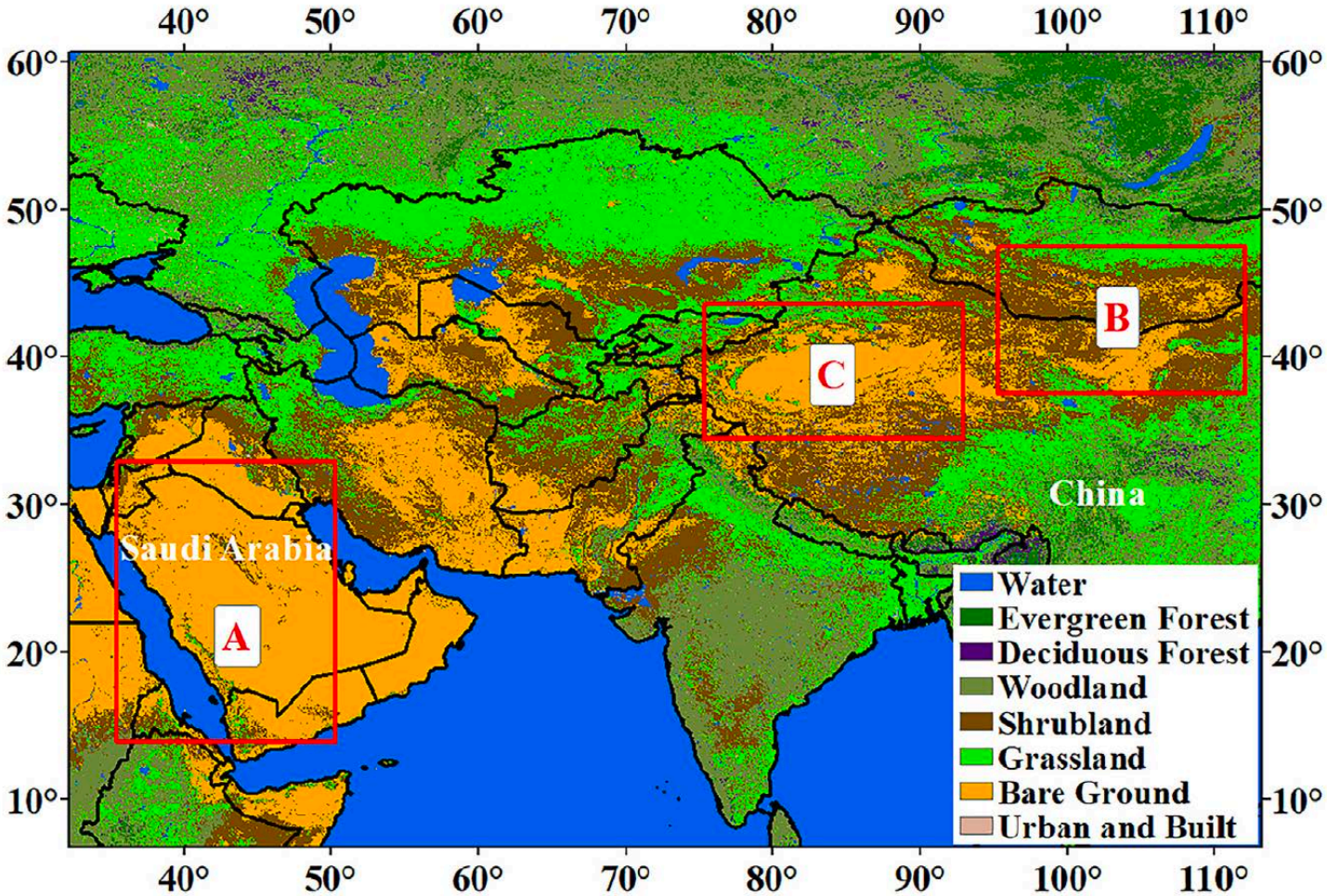



Fig. 1. Study area. (A) Arabian Desert. (B) Gobi Desert. (C) Taklimakan.

The feature vectors are crucial factors for SVM to exhibit relatively high accuracy. Considering the fact that increasing dimension of feature space without increasing the useful information reduces the accuracy (Shahrisvand and Akhoondzadeh, 2013), some spectra and several spectral combinations obtained from some traditional methods were enlisted as the candidate feature vectors. They were B3, B6, B7, B20, B29, B31, B32, (B7 − B3)/(B7 + B3), B20 − B31, B29 − B31, B20 − B29, B32 − B31, and B31/B32 of MODIS L1. The spectral range of each band is shown in Table 3. A trial-and-error procedure was conducted to define the optimal spectral combinations that were finally accepted as the feature vectors of SVM, i.e., we applied different spectral combinations with different number of bands to the SVM classifier and compared the classification precision to find the relatively best spectral combination. We choose the radial basis function (RBF) to perform the classification. The trial-and-error procedure demonstrated that the variation of C and $\gamma$ revealed almost the same results when they were limited in a relatively large range ($C > 0.15$, $\gamma < 0.1$). Consequently, we set the value of regularization parameter (C) and gamma ($\gamma$) in kernel function as commonly used 1.0 and 0.07 respectively, while C (0.25) and $\gamma$ (0.0078) are also utilized in the dust aerosol detection based on the SVM method using CALIPSO data (Ma and Gong, 2012).

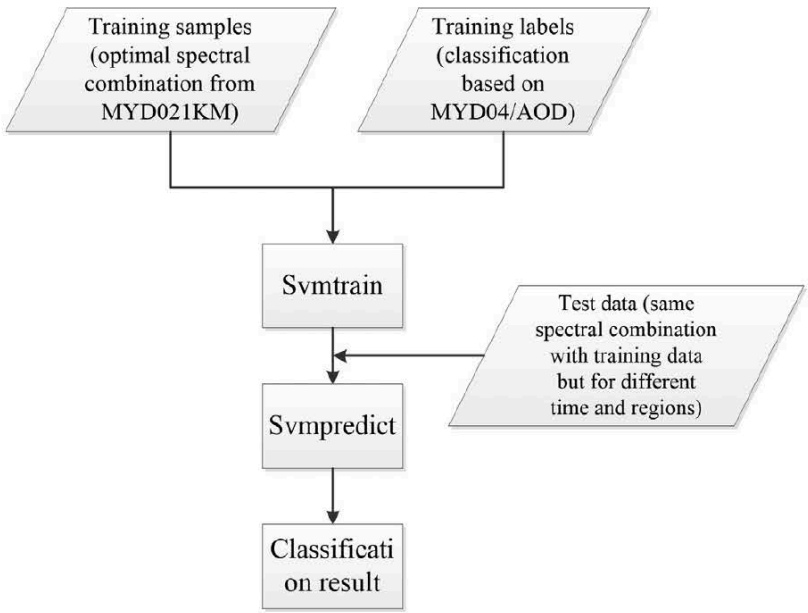



Fig. 2. Flow chart of DSD_SVMS.

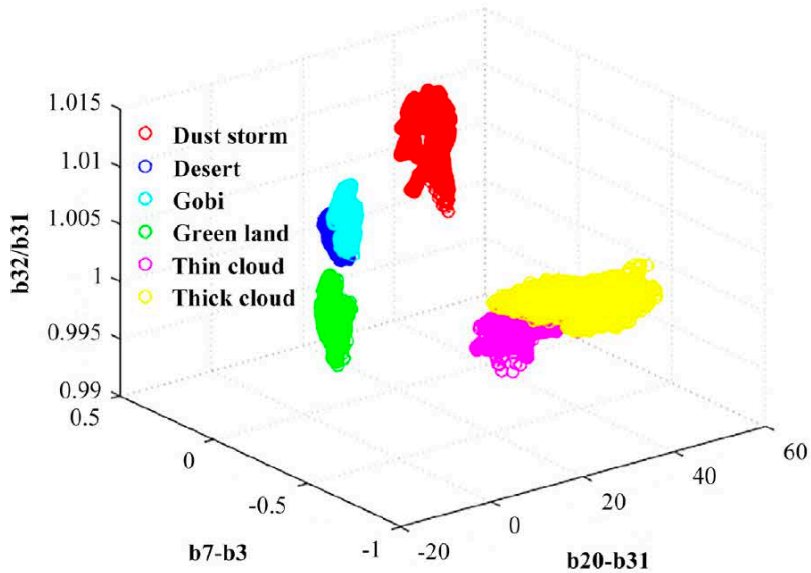



Fig. 3. Clustering characteristic of the combination of (B7 − B3)/(B7 + B3), B20-B31 and B31/B32 in six different objects.

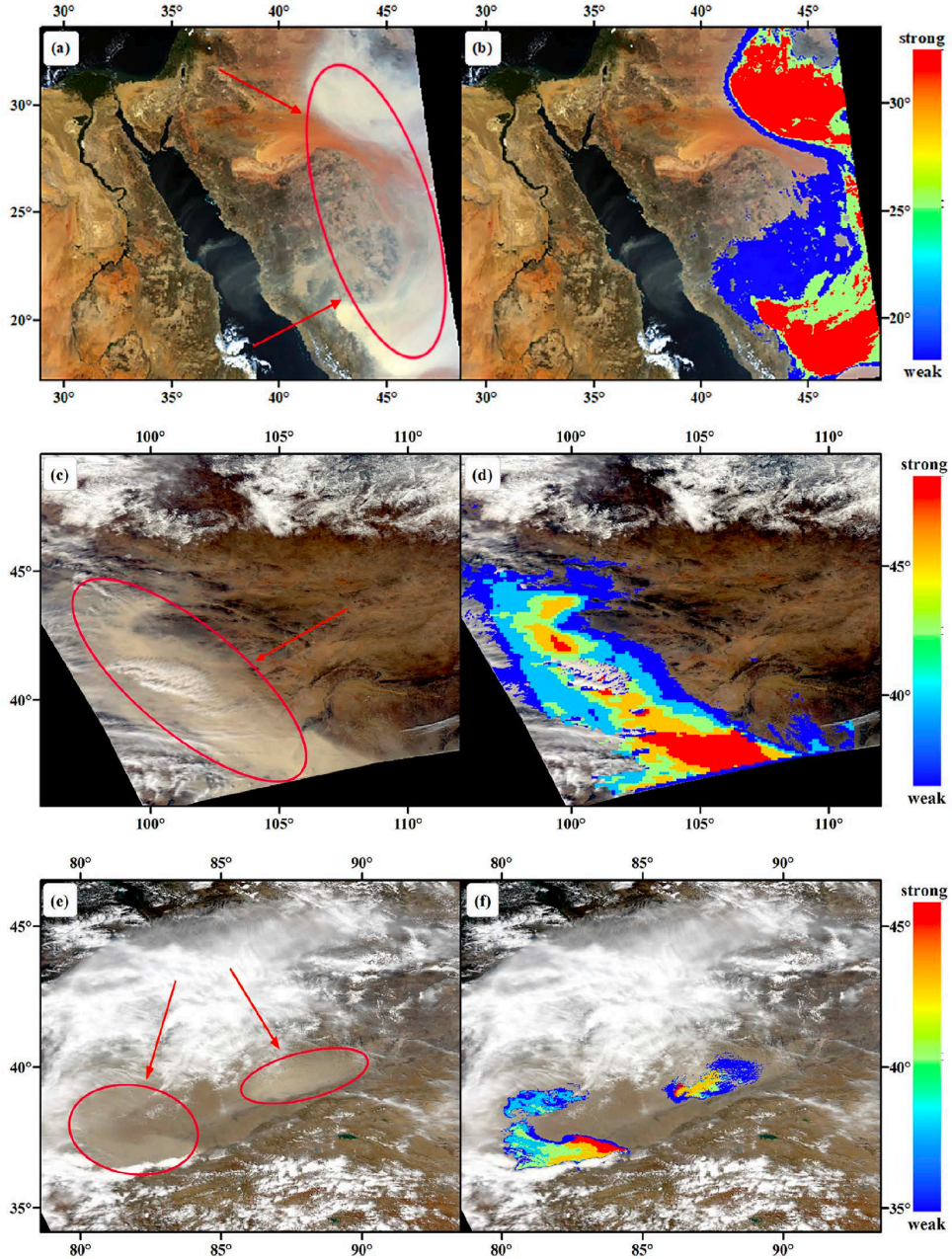



Fig. 4. Comparison of detected dust storms with true color image and SVM results, (a), (c), (e) and (g) are true color for the Arabian Desert on 18 March 2012, Gobi Desert on 9 March 2013, Taklimakan Desert on 23 April 2017 and Gobi Desert on 10 April 2006 respectively; (b), (d), (f) and (h) are for DSD_SVMS results for the same dates and areas respectively.