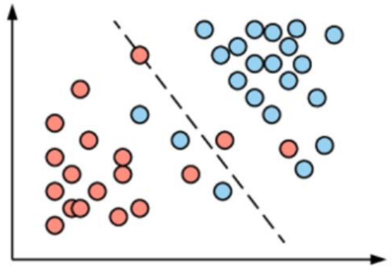
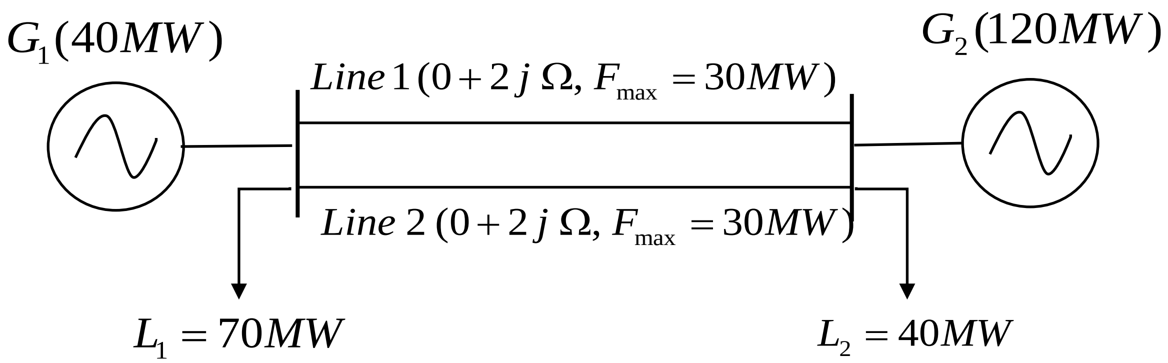
Big Data Technology and its Applications



Logistic regression and Support vector machine

张宁 ningzhang@tsinghua.edu.cn

A problem in power system



- Given a set of operation state and assuming loads are constant, how to judge whether the power system is safe?

ID	G1 generation	G2 generation	Line 1 status	Safe or not
1	0	110	Connected	N
2	20	90	Connected	Y
3	40	70	Connected	Y
4	0	110	Disconnected	N
5	20	90	Disconnected	N
6	40	70	Disconnected	Y

- Lots of similar problems in power systems.

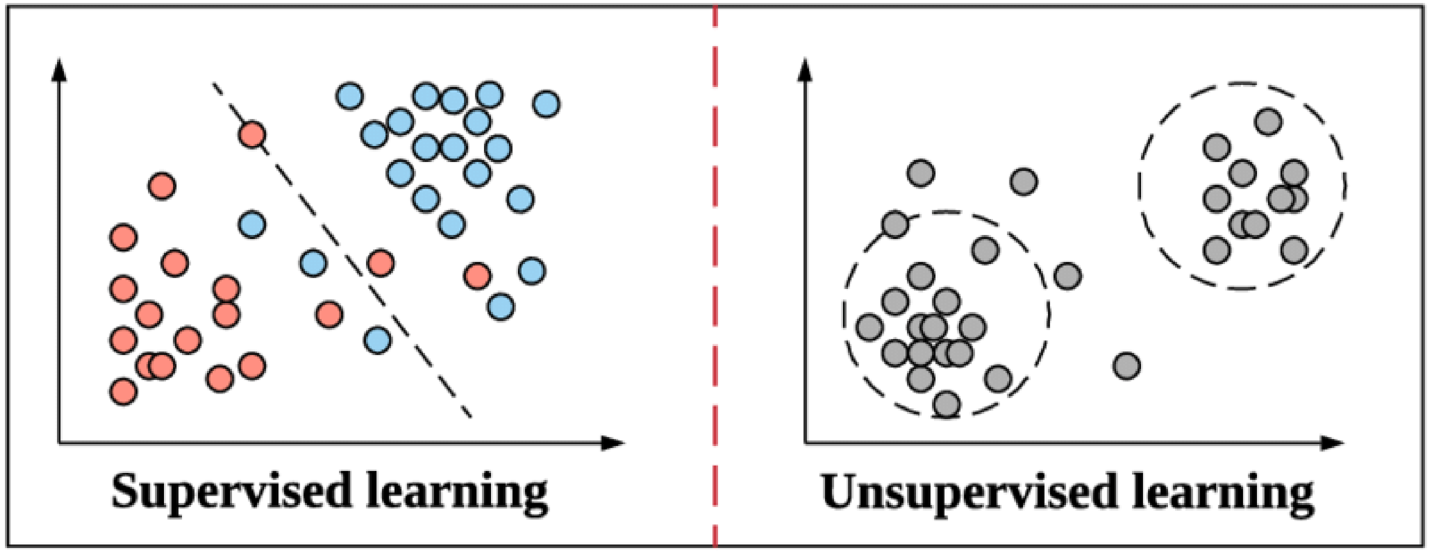
Last class in supervised learning…

- **Classification**: y is discrete. To simplify, $y \in \{-1, +1\}$

$$f : R^d \rightarrow \{-1, +1\}$$

f is called a binary classifier.

- Example: Approve credit yes/no, spam/ham, banana/orange.



- **Methods: Logistic Regression, Support Vector Machines**, neural networks, decision trees, etc.

Logistic regression

What does “logistic” mean?

- **Logistic function:** A logistic function or logistic curve is a common S-shaped curve (sigmoid curve) with equation:

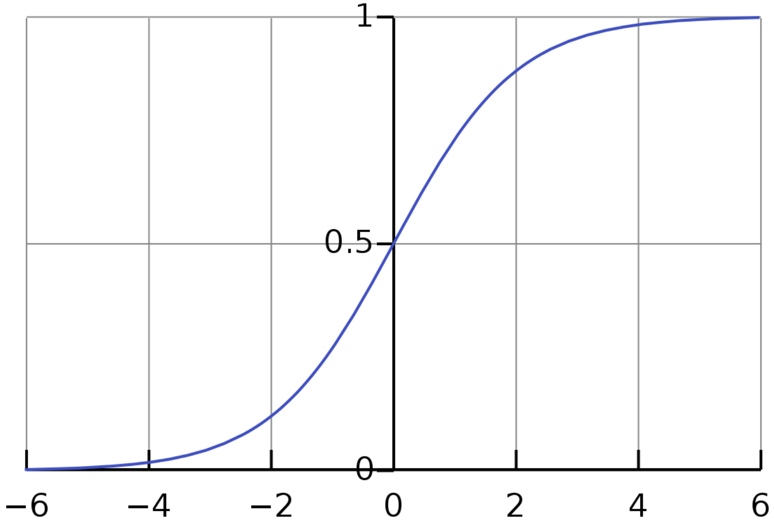
$$f(x) = \frac{L}{1+e^{-kx}}$$

L = the curve's maximum value,
k = the logistic growth rate or steepness of the curve

- **A little history:** The logistic function was introduced in a series of three papers by *Pierre François Verhulst* between 1838 and 1847, who devised it as a model of **population growth**.

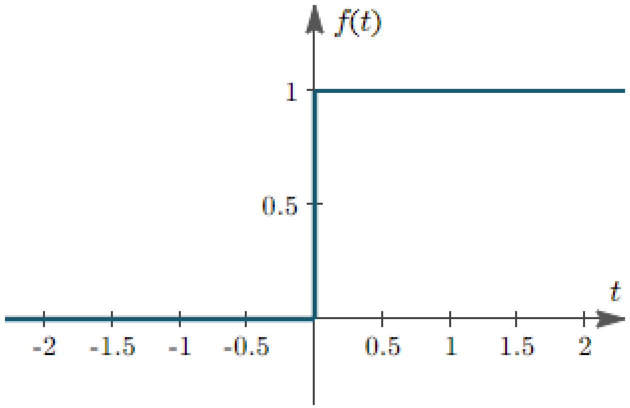
$L = 1, k = 1$:
Standard logistic function

$$f(x) = \frac{1}{1+e^{-x}}$$



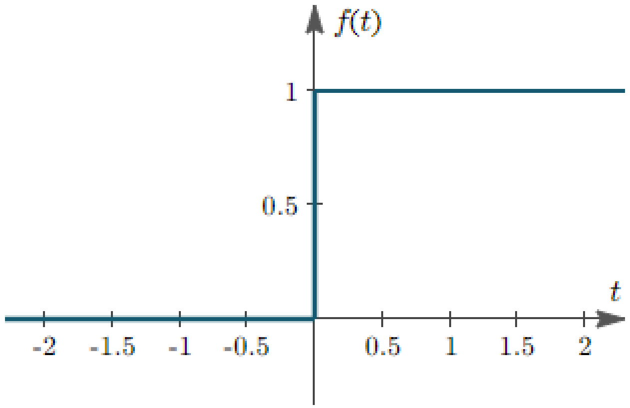
“Logistic” and classification?

- **Ideal classification model**: unit step function

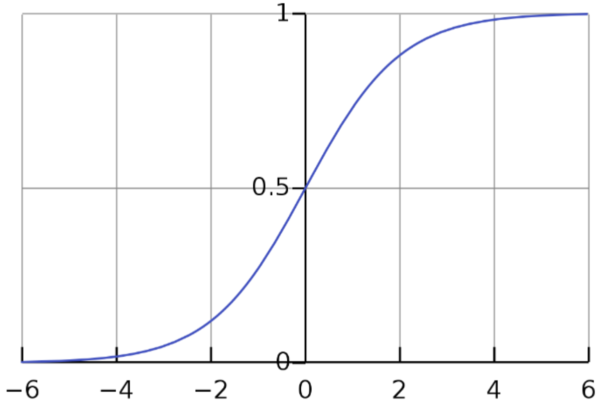


$$y = \begin{cases} 0, & z < 0; \\ 0.5, & z = 0; \\ 1, & z > 0, \end{cases}$$

- However, the unit step function is non-differentiable



logistic function



Logistic regression

- Given input feature $\boldsymbol{x}$ and label $\boldsymbol{y}$, we use the logistic function to make regression:

$$y = \frac{1}{1 + e^{-\left(\boldsymbol{w}^T \boldsymbol{x} + \boldsymbol{b}\right)}}$$

- Can also be written as:

$$\ln \frac{y}{1-y} = \boldsymbol{w}^T \boldsymbol{x} + b$$

- If we view $\boldsymbol{y}$ as the probability, then $y/(1-y)$ can be viewed as **the ratio of the probability of success and the probability of failure**, which represents the likelihood that success will occur.

- $y/(1-y)$ is called **odds.** $\ln(y/(1-y))$ is called **log odds** or **logit**

Logistic regression

- Given a set of training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{i=m}$, how to obtain the optimal parameters $\boldsymbol{w}, b$ in logistic model?
- Recall that If we view $\boldsymbol{y}$ as the probability, then $y/(1-y)$ can be viewed as the ratio of the probability of success and the probability of failure.

$$\ln \frac{y}{1-y} = \boldsymbol{w}^T \boldsymbol{x} + b \quad \Rightarrow \quad \ln \frac{p(y=1 \mid \boldsymbol{x})}{p(y=0 \mid \boldsymbol{x})} = \boldsymbol{w}^T \boldsymbol{x} + b$$

- By applying the maximum likelihood method, we try to maximize the overall probability on the training dataset.

$$\max \sum_{i=1}^{m} \ln p(y_i \mid \boldsymbol{x}_i; \boldsymbol{w}, b)$$

Logistic regression

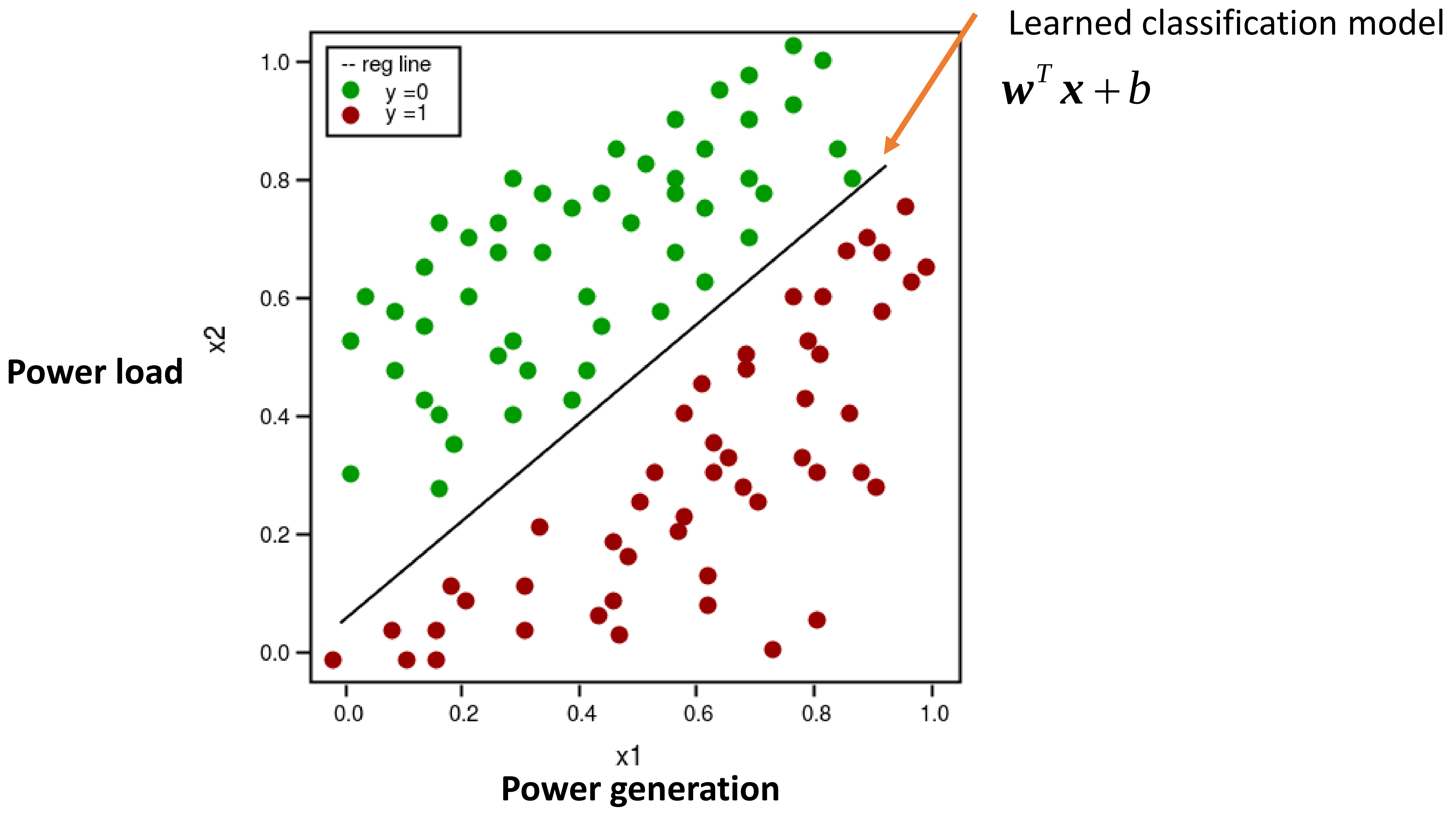
- Let $\beta = (\boldsymbol{w}; b), x = (\boldsymbol{x}; 1)$, then $\boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{b}$ can be simplified as $\boldsymbol{\beta}^T\boldsymbol{x}$.
- Let $p_1 = p(y = 1) = \frac{e^{\boldsymbol{w}^T\boldsymbol{x}+b}}{1+e^{\boldsymbol{w}^T\boldsymbol{x}+b}}$, $\quad p_0 = p(y = 0) = \frac{1}{1+e^{\boldsymbol{w}^T\boldsymbol{x}+b}}$
- Then $p(y_i|\boldsymbol{x}_i; \boldsymbol{w}, b) = y_i p_1 + (1 - y_i)p_0$

maximum likelihood method

$$
\begin{aligned}
\max \ & \sum_{i=1}^{m} \ln p\left(y_i \mid \boldsymbol{x}_i; \boldsymbol{w}, b\right) \\
&= \sum_{i=1}^{m} \ln\left(y_i p_1 + (1-y_i) p_0\right) \\
&= \sum_{i=1}^{m} \ln\left(y_i \frac{e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}}{1+e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}} + (1-y_i)\frac{1}{1+e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}}\right) \\
&= \sum_{i=1}^{m} \ln\left(\frac{y_i\left(e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}-1\right)+1}{1+e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}}\right) \\
&= \sum_{i=1}^{m} \ln\left(y_i\left(e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}-1\right)+1\right) - \ln\left(1+e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}\right)
\end{aligned}
$$

$$
\Rightarrow \quad \max \sum_{i=1}^{m}\left(y_i \boldsymbol{\beta}^T \boldsymbol{x}_i - \ln\left(1+e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}\right)\right)
$$

Logistic regression



Classification Evaluation Metrics

- **Confusion Matrix (混淆矩阵)**:

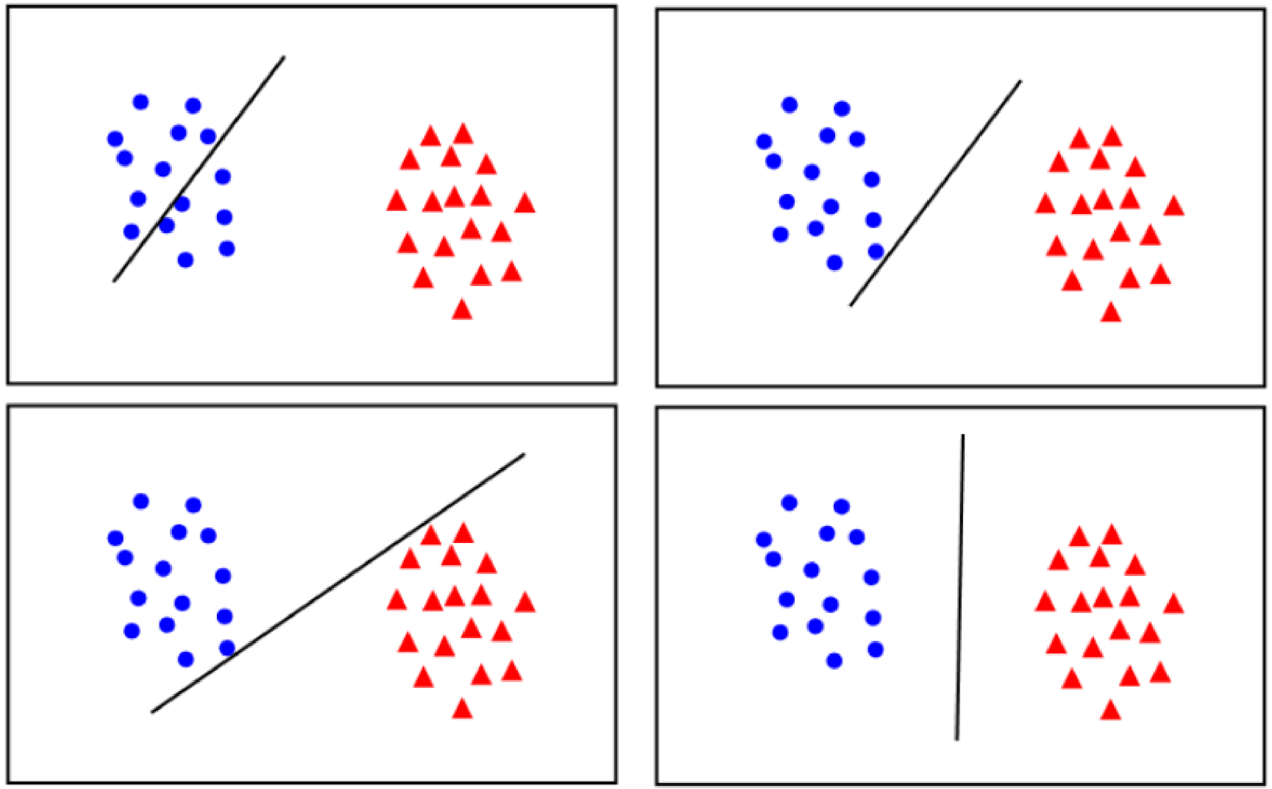
		Actual Label	
		Positive	Negative
Predicted Label	Positive	**True Positive (TP)**	**False Positive (FP)**
	Negative	**False Negative (FN)**	**True Negative (TN)**

Accuracy	(TP + TN) / (TP + TN + FP + FN)	The percentage of predictions that are correct
Precision	TP / (TP + FP)	The percentage of positive predictions that are correct
Sensitivity (Recall)	TP / (TP + FN)	The percentage of positive cases that were predicted as positive
Specificity	TN / (TN + FP)	The percentage of negative cases that were predicted as negative

Support Vector Machine

Motivation

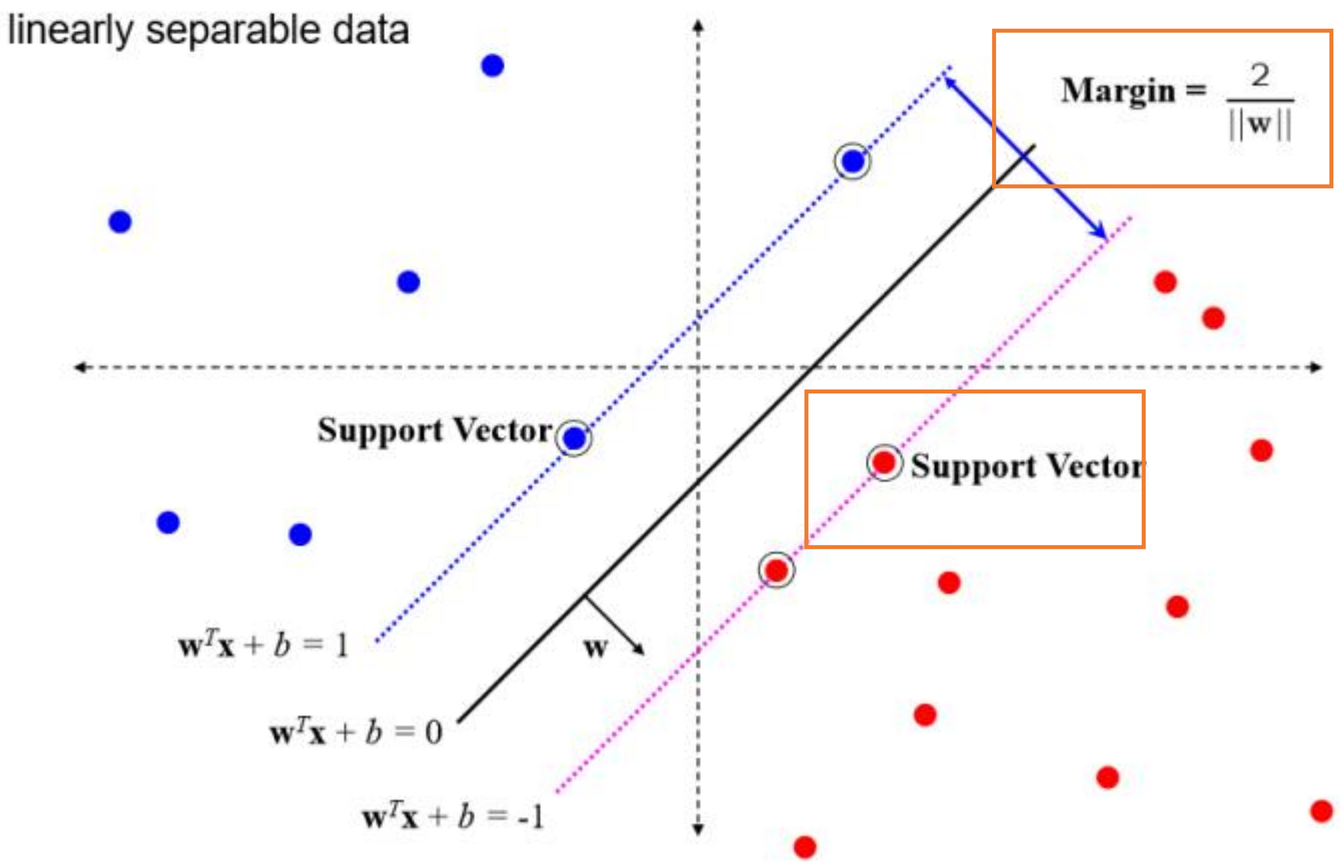
- Which one is the best model for classification?



Best robustness & generalization

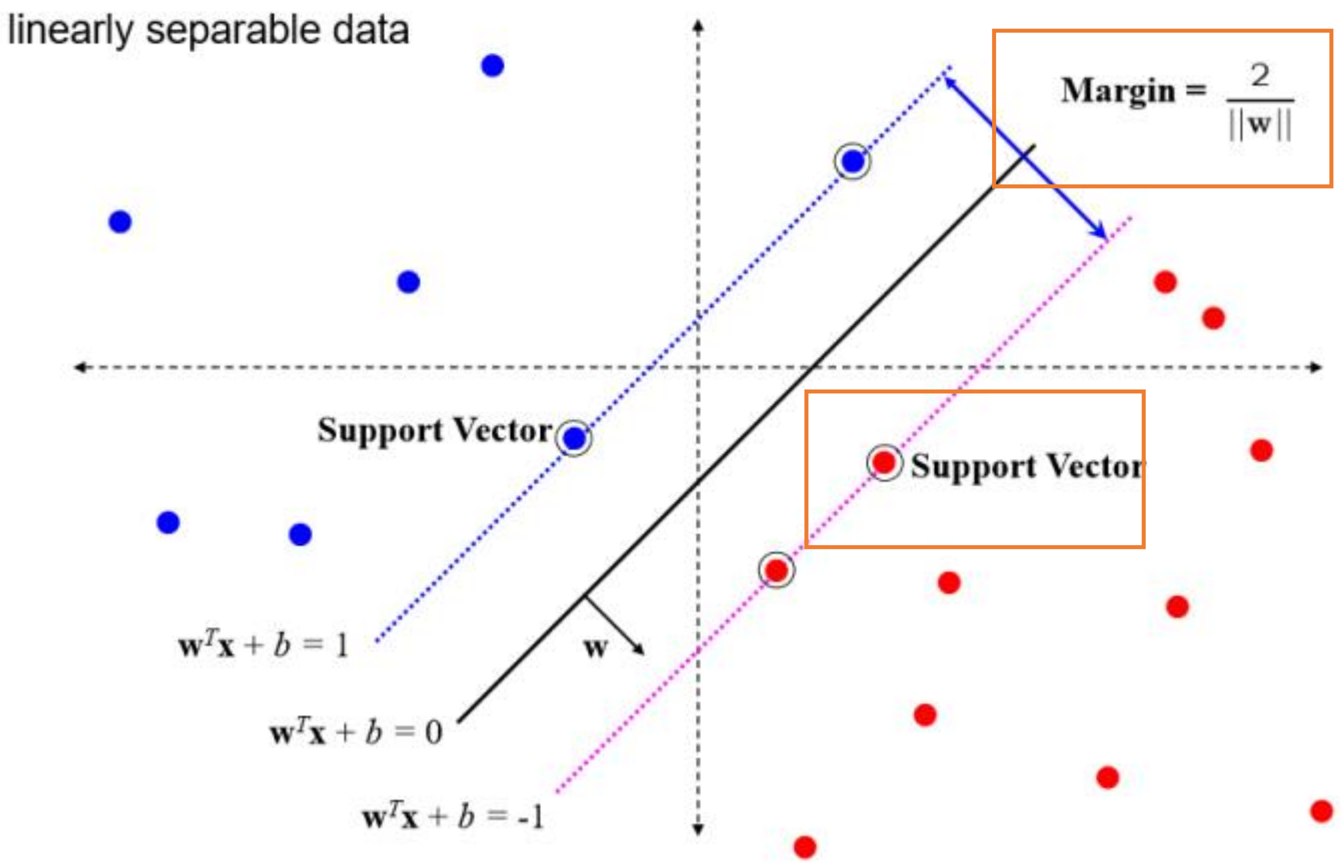
Margin and support vector

- Support Vectors: Input vectors that just touch the boundary of the margin



Support Vector Machine

- Objective: Find the $\boldsymbol{w}$ and b that leads to the largest margin



Support Vector Machine (SVM)

- Given a set of training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{i=m}$, how to obtain the optimal parameters $\boldsymbol{w}, b$ in SVM?
- 1) correctly classifying all training data:

$$\begin{aligned} &\boldsymbol{w}^T \boldsymbol{x}_i + b \ge 1, \quad \text{if} \quad y_i = +1 \\ &\boldsymbol{w}^T \boldsymbol{x}_i + b \le -1, \text{if} \quad y_i = -1 \end{aligned}$$

- 2) maximize the margin

$$\text{margin} = \frac{2}{\|\boldsymbol{w}\|}$$

$$\begin{aligned} \max \quad & \frac{2}{\|\boldsymbol{w}\|} \\ s.t. \quad & y_i\left(\boldsymbol{w}^T \boldsymbol{x}_i + b\right) \ge 1, i = 1, 2, \ldots, m \end{aligned}$$

nonlinear objective

$$\begin{aligned} \min \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ s.t. \quad & y_i\left(\boldsymbol{w}^T \boldsymbol{x}_i + b\right) \ge 1, i = 1, 2, \ldots, m \end{aligned}$$

Convex programming

Support Vector Machine (SVM)

- Recall the KKT condition for solving convex programming
- Given general problem

$$\min f(\boldsymbol{x})$$

$$g_i(\boldsymbol{x}) \leq 0, \quad i = 1, 2, \ldots, m$$

$$h_j(\boldsymbol{x}) = 0, \quad j = 1, 2, \ldots, p$$

$$\boldsymbol{x} \in X \subset \mathbb{R}^n, \boldsymbol{x} = (x_1, x_2, \ldots, x_n)$$

- The Karush-Kuhn-Tucker conditions or KKT conditions are:

$$\frac{\partial L(x,u,v)}{\partial x} = 0$$ stationarity

$$\mu \bullet g(x) = 0$$ complementary slackness

$$g(x) \leq 0, h(x) = 0$$ primal feasibility

$$\mu \geq 0$$ dual feasibility

Support Vector Machine (SVM)

- Step 1: generate the Lagrangian function

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \alpha_i \left(1 - y_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + b\right)\right)$$

- Step 2: let $\dfrac{\partial L(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial \boldsymbol{w}} = 0, \dfrac{\partial L(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial b} = 0$

$$\Rightarrow \quad \boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \qquad \sum_{i=1}^{m} \alpha_i y_i = 0$$

- Step 3: Take the above equation into the original optimization problem:

$$\begin{aligned} &\max \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \\ &s.t. \quad \sum_{i=1}^{m} \alpha_i y_i = 0, \alpha_i \geq 0 \end{aligned}$$

Support Vector Machine (SVM)

- Step 4: Take $\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i$ into the equation, we get the final SVM model

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + b$$

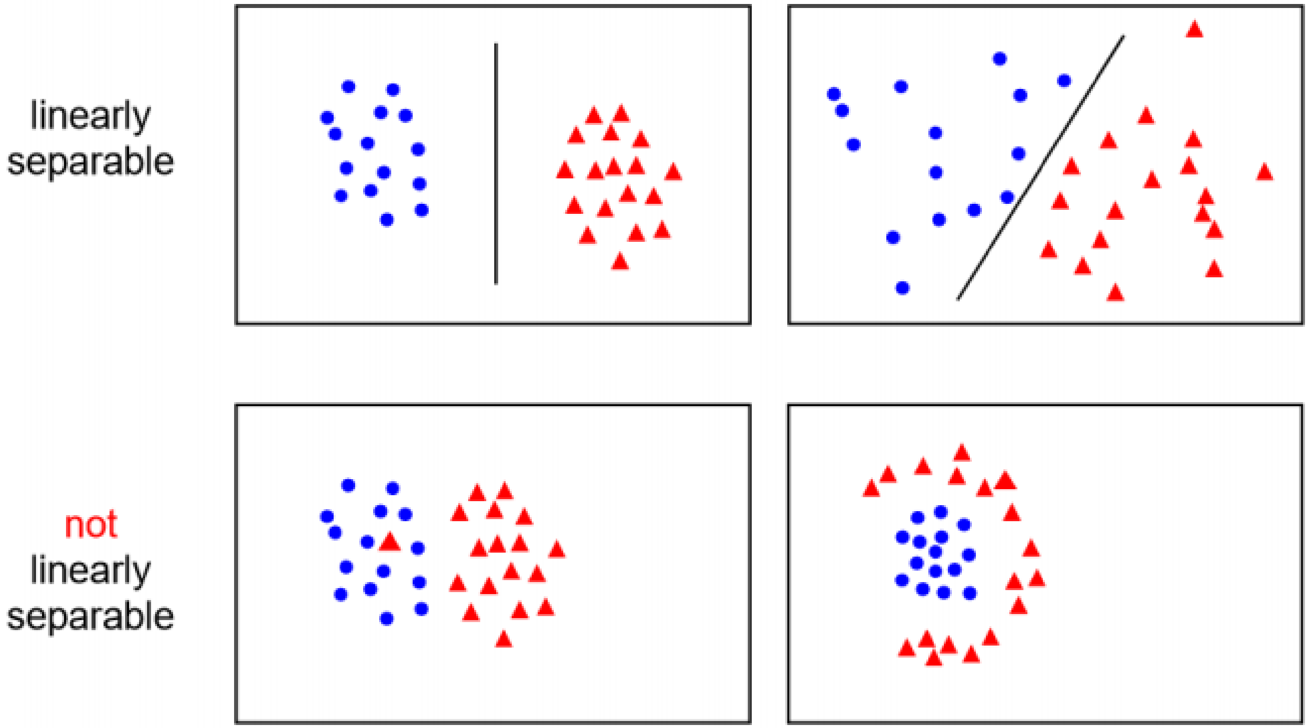
- From KKT condition we know:

$$\begin{cases} \alpha_i \geq 0 \\ y_i f(\boldsymbol{x}_i) \geq 1 \\ \alpha_i \left(y_i f(\boldsymbol{x}_i) - 1 \right) = 0 \end{cases} \Rightarrow \quad \alpha_i = 0 \quad \text{or} \quad y_i f(\boldsymbol{x}_i) = 1$$

- This means that the final model only depends on the boundary data (support vector)------ Support Vector Machine.

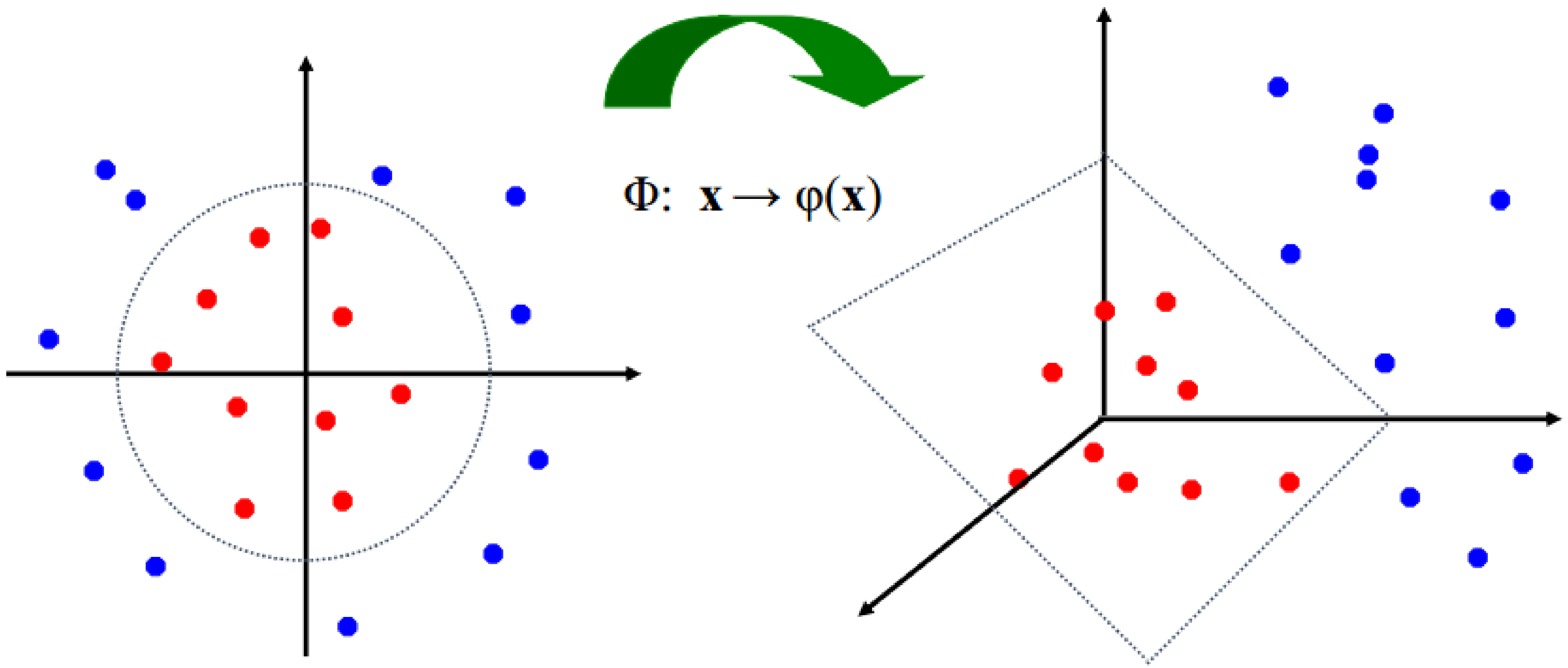
Support Vector Machine (SVM)

- Up to now, we can solve the linearly separable problem using SVM or logistic regression.
- What if the data is not linearly separable?



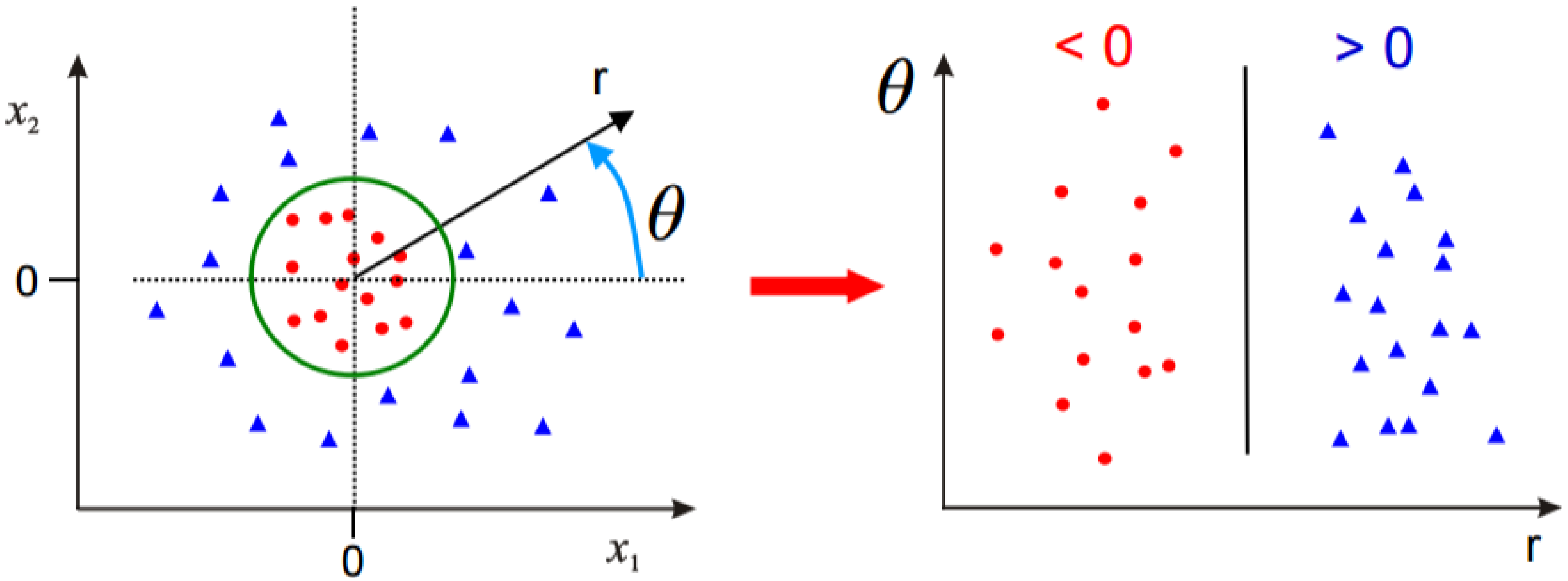
Nonlinear SVM

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Nonlinear SVM

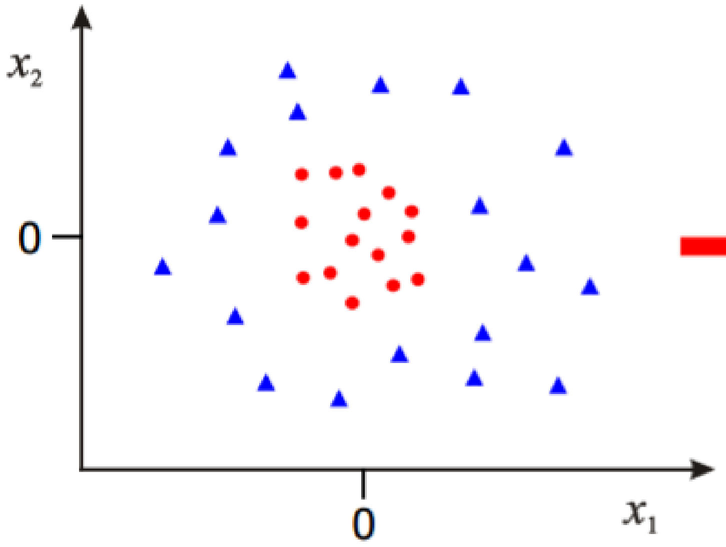
- Example 1: Data is linearly separable in polar coordinates

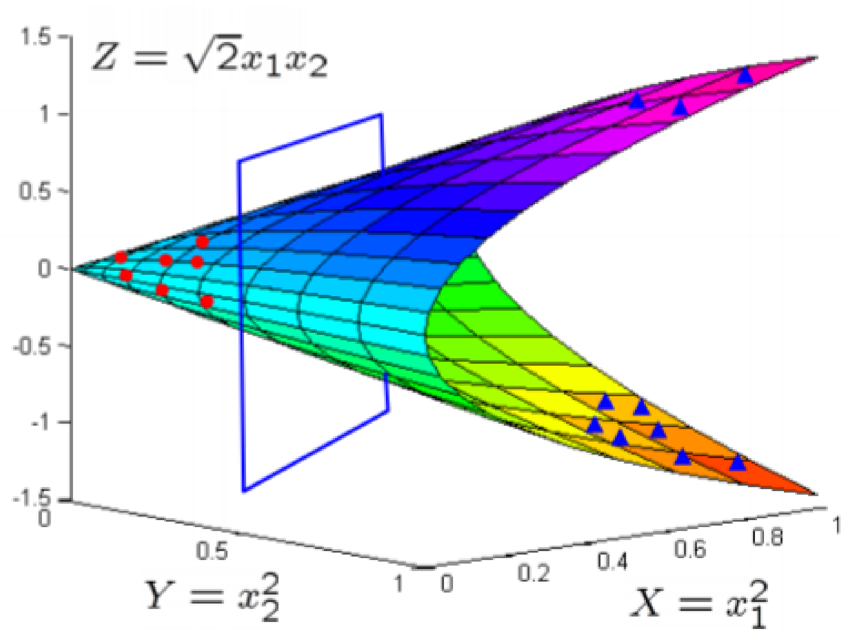


Mapping function $\phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix}$

Nonlinear SVM

- Example 2: Data is linearly separable in 3D

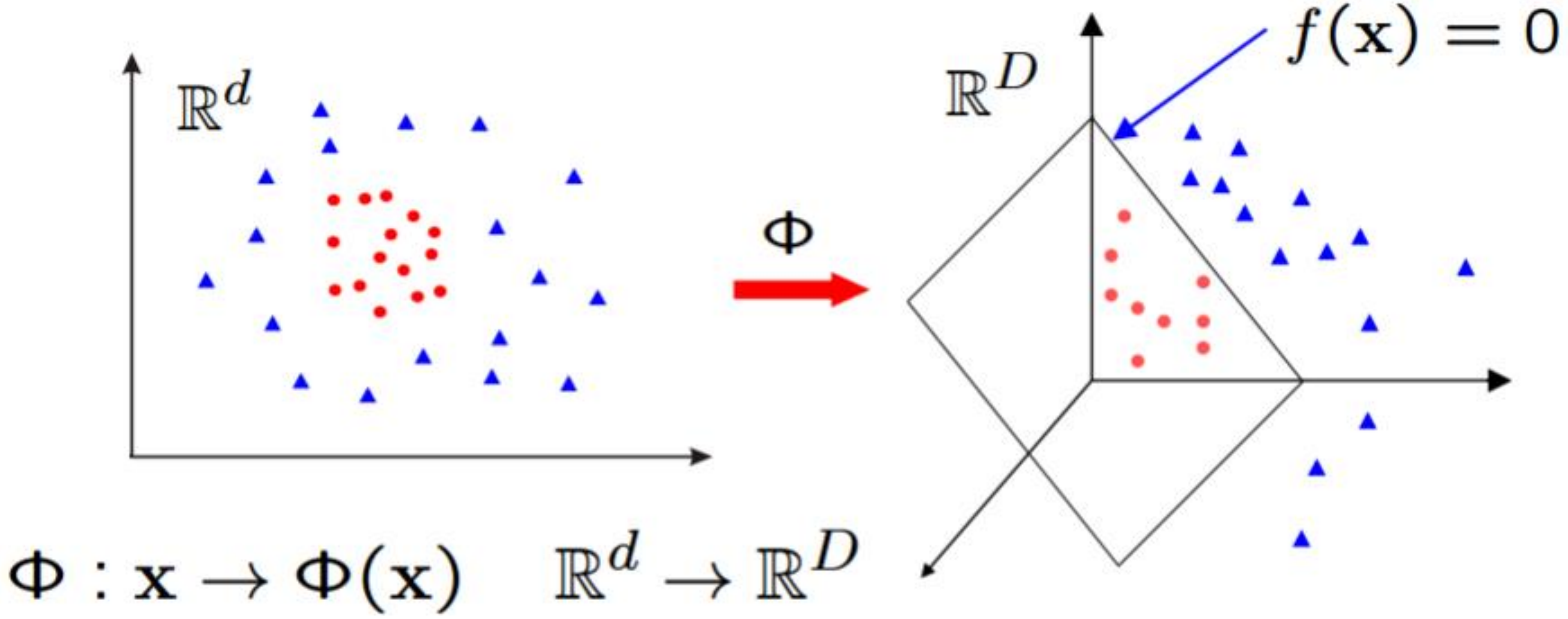




Mapping function $\phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$

Nonlinear SVM

- Learn the classifier at a high-dimensional space to tackle the nonlinearity problem.



$$f(x) = w^T x + b$$

$$f(x) = w^T \phi(x) + b$$

Linear SVM

NonLinear SVM

Nonlinear SVM

- How to learn nonlinear SVM?
- Basic idea: by using the mapping function $x \to \phi(x)$, the data is linear separable regarding to $\phi(x)$.

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\
s.t. \quad & y_i\left(\boxed{\boldsymbol{w}^T\phi(\boldsymbol{x}_i)}+b\right) \geq 1, i=1,2,\ldots,m
\end{aligned}
$$

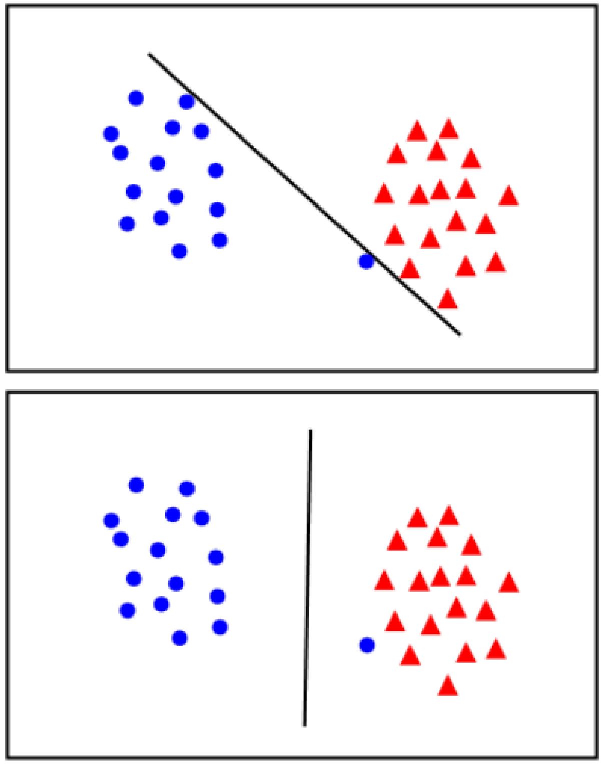
$$
\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{w}^T\phi(\boldsymbol{x})+b=\sum_{i=1}^{m}\alpha_i y_i \boxed{\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x})}+b
$$

- Note that $\phi(x)$ only occurs in pairs $\phi(x_j)^T\phi(x_i)$
- Write $k(x_j, x_i) = \phi(x_j)^T\phi(x_i)$. This is known as a **kernel**.

Useful kernels

Name	Mathematical formulation
Linear kernel	$k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \boldsymbol{x}_i^T \boldsymbol{x}_j$
Polynomial kernel	$k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \left(\boldsymbol{x}_i^T \boldsymbol{x}_j\right)^d$
Gaussian kernel (RBF kernel)	$k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \exp\left(-\frac{\lVert x_i - x_j \rVert^2}{2\sigma^2}\right)$
Laplace RBF kernel	$k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \exp\left(-\frac{\lVert x_i - x_j \rVert}{\sigma}\right)$
Sigmoid kernel	$k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \tanh\left(\beta \boldsymbol{x}_i^T \boldsymbol{x}_j + \theta\right)$

Dataset with noise



• the points can be linearly separated but there is a very narrow margin

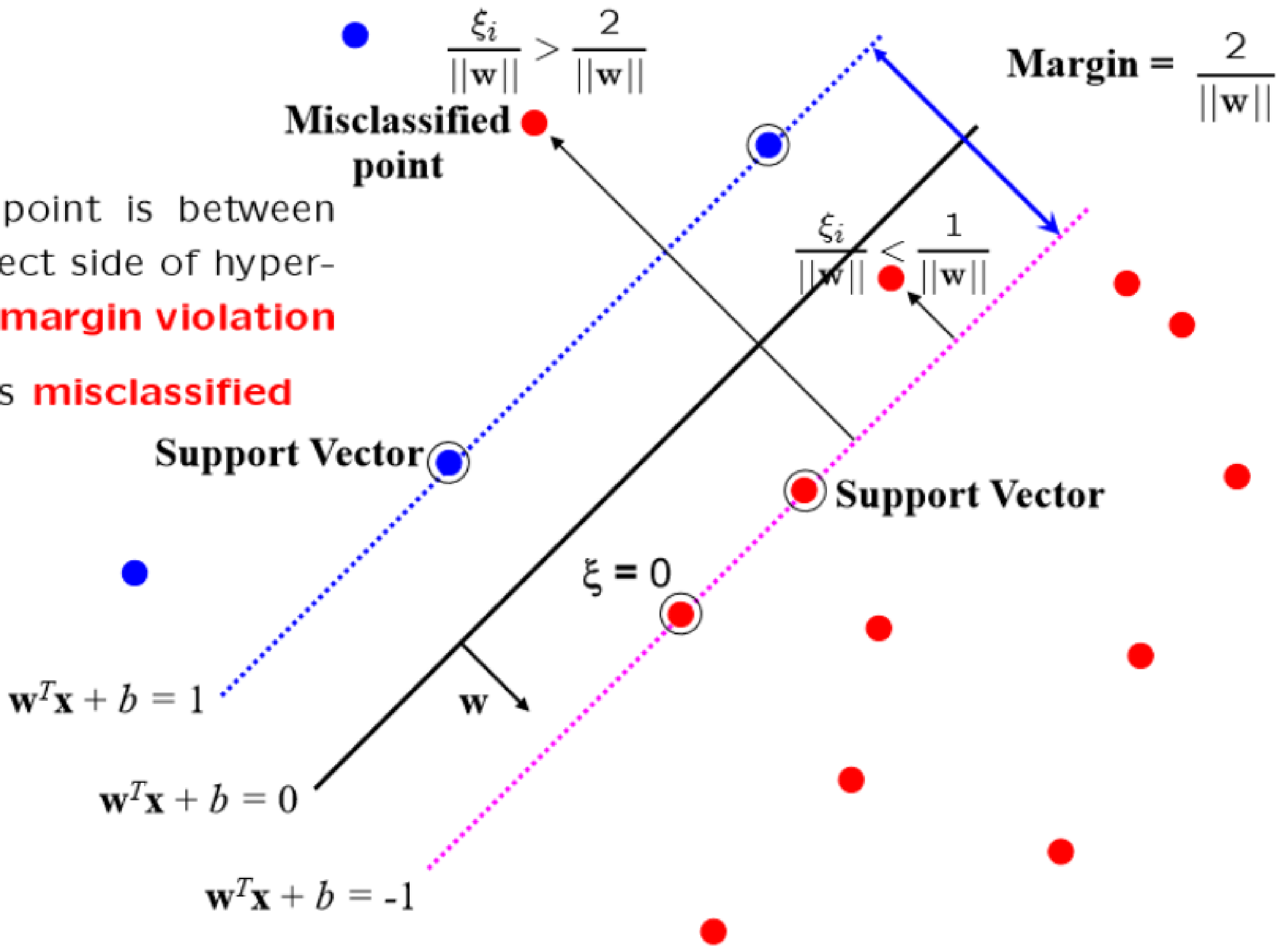
• but possibly the large margin solution is better, even though one constraint is violated

- In general there is a trade off between the margin and the number of mistake on the training data

Introduce slack variables

$$\xi_i \geq 0$$

- for $0 < \xi \leq 1$ point is between margin and correct side of hyper-plane. This is a **margin violation**
- for $\xi > 1$ point is **misclassified**



Using “soft margin”

- **Basic idea**: maximize the margin while minimizing the number of misclassified data.

$$\begin{aligned} \min \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ s.t. \quad & y_i\left(\boldsymbol{w}^T\phi(\boldsymbol{x}_i)+b\right)\geq 1, i=1,2,\dots,m \end{aligned} \quad \Rightarrow \quad \begin{aligned} \min \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ s.t. \quad & y_i\left(\boldsymbol{w}^T\phi(\boldsymbol{x}_i)+b\right)\geq 1-\xi_i, \xi\geq 0, i=1,2,\dots,m \end{aligned}$$

- C is the regularization parameter:

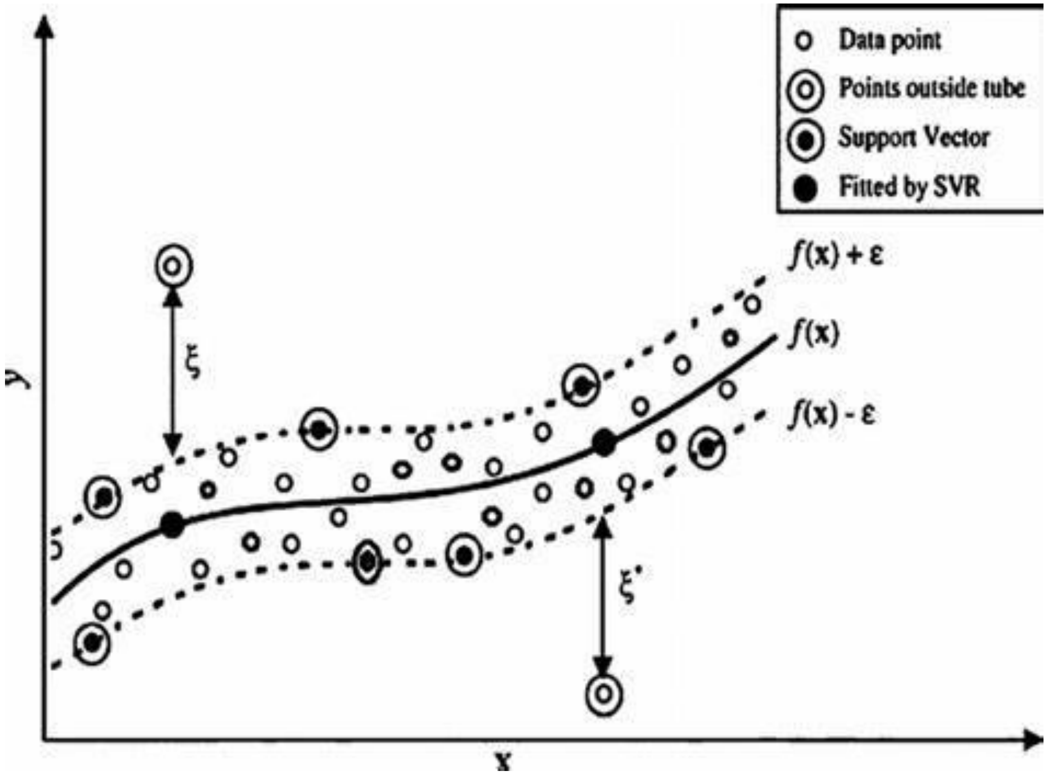
- Small C allows constraints to be easily ignored --------- large margin
- Large C makes constraints hard to ignore ---------- narrow margin
- If C is big enough --------- hard margin

Support Vector Regression (SVR)

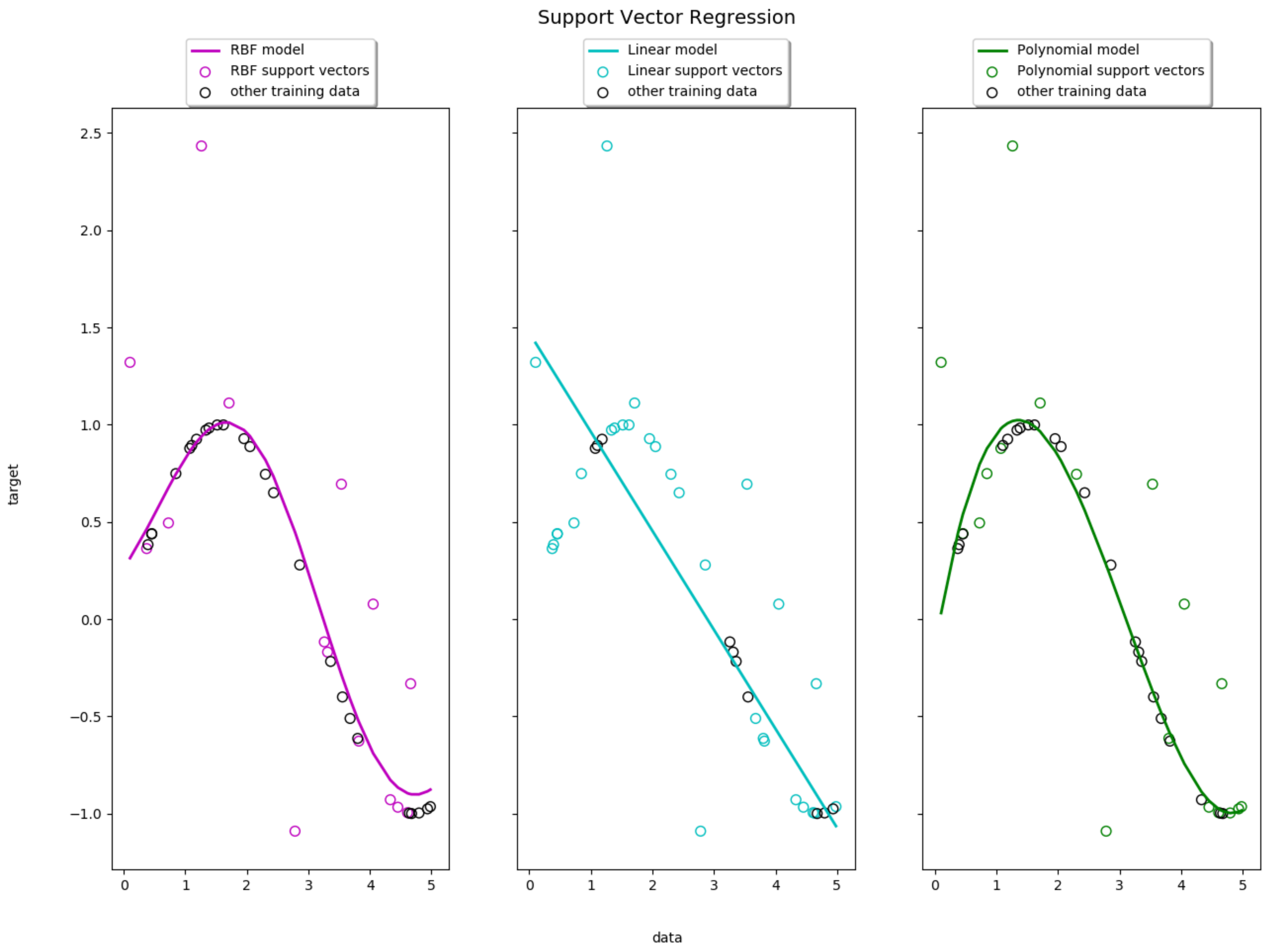
- Try to formulate the mathematical problem by using the idea of "margin"

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\
s.t. \quad & f(\boldsymbol{x}_i) - y_i \le \xi_i + 1 \\
& y_i - f(\boldsymbol{x}_i) \le \xi_i + 1 \\
& \xi_i \ge 0
\end{aligned}
$$

- Similar with the SVM!



Using different kernels in SVR



How to apply SVM

- **LIBSVM:** https://www.csie.ntu.edu.tw/~cjlin/libsvm/
- **scikit-learn :** https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

LIBSVM -- A Library for Support Vector Machines

Chih-Chung Chang and Chih-Jen Lin

Version 3.24 released on September 11, 2019. It conducts some minor fixes.
LIBSVM tools provides **many extensions** of LIBSVM. Please check it if you need some functions not supported in LIBSVM.
We now have a nice page LIBSVM data sets providing problems in LIBSVM format.
A practical guide to SVM classification is available now! (mainly written for beginners)
We now have an easy script (easy.py) for users who know NOTHING about SVM. It makes everything automatic--from data scaling to parameter selection.
The parameter selection tool grid.py generates the following contour of cross-validation accuracy. To use this tool, you also need to install python and gnuplot.