Convex Optimization Theory and Applications

Topic 6 - Classification

Li Li

Department of Automation
Tsinghua University

Fall, 2009-2021.

6.0. Outline

6.0. Outline

6.1. Origins of SVM

6.1.0 Separation of Convex Sets

Definition of a hyperplane: a hyperplane is a set with the form $\left\{x \mid a^T x = b\right\}$, where $a \in R^n$, $a \neq 0$, and $b \in R$. A hyperplane divides R^n into two halfspaces. A weak half space or closed half space contains the hyperlane; while a strict half space or open half space does not contains the hyperlane.

Strong separation theorem and Proper separation theorem

6.1. Origins of SVM

6.1.0 Separation of Convex Sets

Projection of a point onto a finite-dimensional closed convex set

Separation of convex sets in finite-dimensional vector spaces

Support hyperplane theorem

Nonuniqueness of the hyperplanes

6.1. Origins of SVM

6.1.0 Separation of Convex Sets

The hyperplane strictly separates set A and B, if A and B are in disjoint open half spaces.

The hyperplane strongly separates set A and B if A and B are in disjoint closed half spaces. That is, there is a $\varepsilon > 0$ such that $A \subset \left\{ x \mid a^T x \geq b + \varepsilon \right\}$ and $B \subset \left\{ x \mid a^T x \leq b \right\}$ (or vice versa).

Another way to state strong separation is $\inf_{x \in A} a^T x > \sup_{y \in B} a^T y$ (or swap A and B).

6.1. Origins of SVM

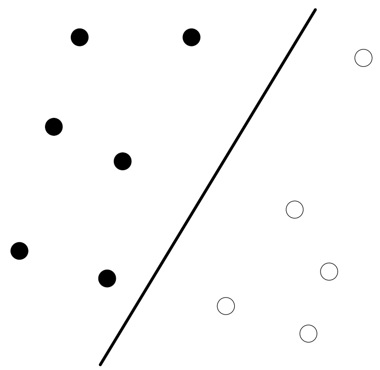
6.1.1 Linear Discrimination

"*The support vector machine (SVM) is the first contact that many optimization researchers had with machine learning, due to its classical formulation as a convex quadratic program - simple in form, though with a complicating constraint. It continues to be a fundamental paradigm today, with new algorithms being proposed for difficult variants, especially large-scale and nonlinear variants. Thus, SVMs offer excellent common ground on which to demonstrate the interplay of optimization and machine learning*."

- *Optimization for Machine Learning*, Chapter 1.

6.1. Origins of SVM

6.1.1 Linear Discrimination



Suppose we want to separate two sets of points (vectors) $\{x_1, \ldots, x_N\}$, $\{x_1, \ldots, x_M\}$ by a hyperplane

$$w^T x_i + b > 0, \quad i = 1, \ldots, N \qquad (6.1)$$

$$w^T x_j + b < 0, \quad j = 1, \ldots, M \qquad (6.2)$$

6.1. Origins of SVM

6.1.1 Linear Discrimination

Eq.(6.1)-(6.2) is equivalent to a set of linear inequalities that are homogeneous in w, b.

We want to further find the optimal hyperplane separates different classes with maximal margin (the distance between the hyperplane and the closest training data point). Such goal can be defined as maximization of the minimum distance between points and the hyperplane

$$\max_{w,b} \ \min\left\{ \|x - x_i\| : w^T x + b = 0, i = 1, \ldots, M + N \right\} \qquad (6.3)$$

6.1. Origins of SVM

6.1.1 Linear Discrimination

The parameters w and b can be rescaled in such a way that the point closest to the hyperplane $w^T x + b = 0$ lies on two hyperplanes $w^T x_i + b = \pm 1$. Thus, we can classify by calculating

$$w^T x_i + b \geq 1, \quad i = 1, \ldots, \quad N \tag{6.4}$$

$$w^T x_j + b \leq -1, \quad j = 1, \ldots, \quad M \tag{6.5}$$

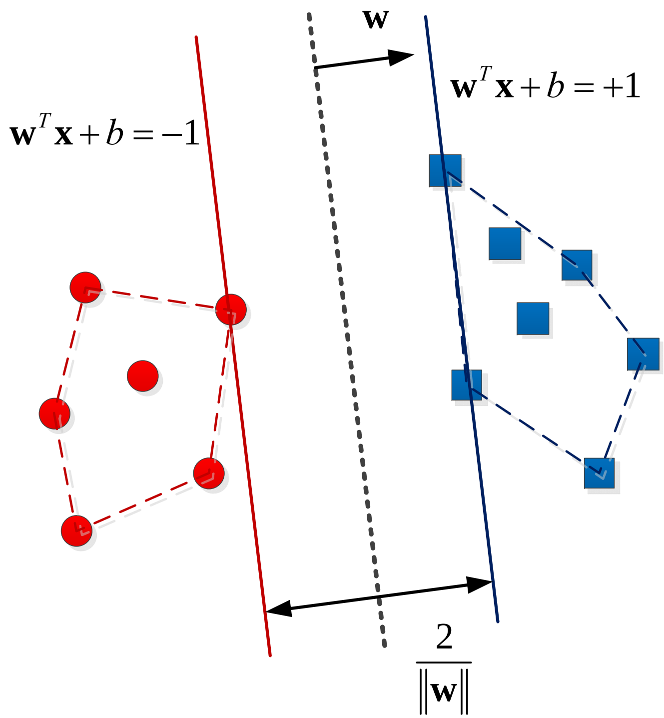
The problem (6.3) can then be rewritten as

$$\min_{w,b} \frac{1}{2} \|w\|_2 \tag{6.6}$$

6.1. Origins of SVM

6.1.1 Linear Discrimination

Please prove that the Euclidean distance between hyperplanes $H_1 = \{z \mid w^T z + b_i = 1\}$ and $H_2 = \{z \mid w^T z + b_i = -1\}$ equals $2/\|w\|_2$.



6.1. Origins of SVM

6.1.1 Linear Discrimination

Given any two points z_1 and z_2 satisfying

$$w^T z_1 + b_i = 1$$

$$w^T z_2 + b_i = -1$$

We have

$$w^T (z_1 - z_2) = 2$$

The shortest distance is

$$\frac{w^T}{\|w\|}(z_1 - z_2) = \frac{2}{\|w\|}$$

6.1. Origins of SVM

6.1.1 Linear Discrimination

So, the idea of linear discrimination is then to separate two sets of points by maximum margin

$$\min_{w,b} \frac{1}{2}\|w\|^2 \tag{6.7}$$

subject to $w^T x_i + b \geq 1$, $w^T x_j + b \leq -1$.

Clearly, this is a quadratic programming in w, b, with linear constraints.

Indeed, this is the primal form of the problem.

6.1. Origins of SVM

6.1.1 Linear Discrimination

We can introduce the sign variable y_i and y_j as follows:

$$y_i = 1 \text{ for } w^T x_i + b \geq 1, \quad i = 1, \ldots, N \qquad (6.8)$$

$$y_j = -1 \text{ for } w^T x_j + b \leq -1, \quad j = 1, \ldots, M \qquad (6.9)$$

Thus, we can rewrite (6.5) into a uniform constraint

$$y_k \left(w^T x_k + b \right) \geq 1, \quad k = 1, \ldots, l = N + M \qquad (6.10)$$

6.1. Origins of SVM

6.1.1 Linear Discrimination

Thus, the classifications function is

$$f(x,w,b) = \mathrm{sgn}(w^T x + b) \tag{6.11}$$

If chosen a point z, we have $f(z,w,b) = \mathrm{sgn}(w^T x + b) = 1$, it should be the first class; otherwise, if $f(z,w,b) = -1$, it should be the second class.

Due to the high dimensionality of the vector variable w, we usually solve it through its Lagrangian function

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i (y_i \cdot ((x_i \cdot w) + b) - 1) \tag{6.12}$$

6.1. Origins of SVM

6.1.1 Linear Discrimination

According to the Lagrangian dual method, we have

$$\frac{\partial}{\partial b}L(w,b,\alpha)=0, \quad \frac{\partial}{\partial w}L(w,b,\alpha)=0 \qquad (6.13)$$

Primal variables vanish at

$$\sum_{i=1}^{l}a_i y_i = 0, \quad w=\sum_{i=1}^{l}\alpha_i y_i x_i \qquad (6.14)$$

6.1. Origins of SVM

6.1.1 Linear Discrimination

The Karush-Kuhn-Thucker KKT condition is

$$\alpha_i \cdot [y_i((x_i \cdot w) + b) - 1] = 0, \quad i = 1, \ldots, \quad l = N + M \qquad (6.15)$$

Clearly, the non-zero α_i correspond to $y_i((x_i \cdot w) + b) = 1$. It means that the vectors which lie on the margin play the crucial role in the solution of the optimization problem. Such vectors are called support vectors.

6.1. Origins of SVM

6.1.1 Linear Discrimination

The α_k terms constitute a dual representation for the weight vector in terms of the training set

$$\hat{w} = \sum_{k=1}^{l} \alpha_k y_k x_k \tag{6.16}$$

This also indicates that the maximum margin hyperplane and therefore the classification task is only a function of the support vectors.

What the dual form should be?

6.1. Origins of SVM

6.1.1 Linear Discrimination

Some main reasons mentioned for solving in the dual:

1. The duality theory provides a convenient way to deal with the constraints

2. We may use possible less computation costs, due to the assumption of supporting vectors

3. The dual optimization problem can be written in terms of dot products, thus making it possible to use kernel functions (Do not understand? No problem, we will explain in the below)

6.1. Origins of SVM

6.1.1 Linear Discrimination

The Lagrangian dual form can then be written as

$$\max_{\alpha} \sum_{p=1}^{l} \alpha_p - \frac{1}{2} \sum_{p=1}^{l} \sum_{q=1}^{l} \alpha_p \alpha_q y_p y_q x_p^T x_q \qquad (6.17)$$

subject to $\sum_{k=1}^{l} \alpha_k y_k = 0$ (Complementary slackness) and $\alpha_k \geq 0$ (Dual feasibility).

Usually, we also require $0 \leq \alpha_k \leq C$ to control the balance between training accuracy and the margin width.

6.1. Origins of SVM

6.1.1 Linear Discrimination

Because the weight is $\hat{w} = \sum_{k=1}^{l} \alpha_k y_k x_k$, if C is very small, all the α_k will be very small, and we may be unable to get values of $\hat{w}$ bigger than 1. This will manifest itself as a solution that has no unbounded support vectors, and an inability to calculate b. So, if find no feasible solution, we need to make C bigger and try again.

In general, to find the best parameter C, we have to test different pre-settings and compare their performance on an independent data set, which is known as validation.

6.1. Origins of SVM

6.1.1 Linear Discrimination

C controls the tradoff between samll function norm and empirical risk minimization. We do not want C to be too small or too large.

If C is very large, we are saying we want good training error, and do not care too much about the norm of the function. Espeically, when the problem is hard, the norm can get very large, which can lead to poor generalization.

Additionally, larger C will in turn imply the possibility of a long training time. These two phenomena are closely linked. Very long training times are often a sign of overfitting and poor generalization error. So, we often try to make C smaller.

6.1. Origins of SVM

6.1.1 Linear Discrimination

For simplicity reasons, we often require that the hyperplane passes through the origin of the coordinate system.

Such hyperplanes are called unbiased hyperplanes, whereas others are called biased. An unbiased hyperplane can be enforced by setting $b = 0$ in the primal optimization problem. The corresponding dual is identical to the dual given above without the equality constraint (the vanish condition)

$$\sum_{l=1}^{l} \alpha_k y_k = 0 \tag{6.18}$$

6.1. Origins of SVM

6.1.1 Linear Discrimination

Reasons for unbiased SVM, Steinwart, Hush, Scovel [17]:

(1) The generalization performance of SVMs for classification does not suggest that the offset offers any improvement for such kernels.

(2) SVM optimization problem with offset imposes more restrictions on solvers than the optimization problem without offset does. For example, the offset leads to an additional equality constraint in the dual optimization problem, which in turn makes it necessary to update at least two dual variables at each iteration of commonly used solvering algorithm.

6.1. Origins of SVM

6.1.2 Robust Linear Discrimination

If data are not linearly separable:

Primal problem is
Dual problem is

6.1. Origins of SVM

6.1.2 Robust Linear Discrimination

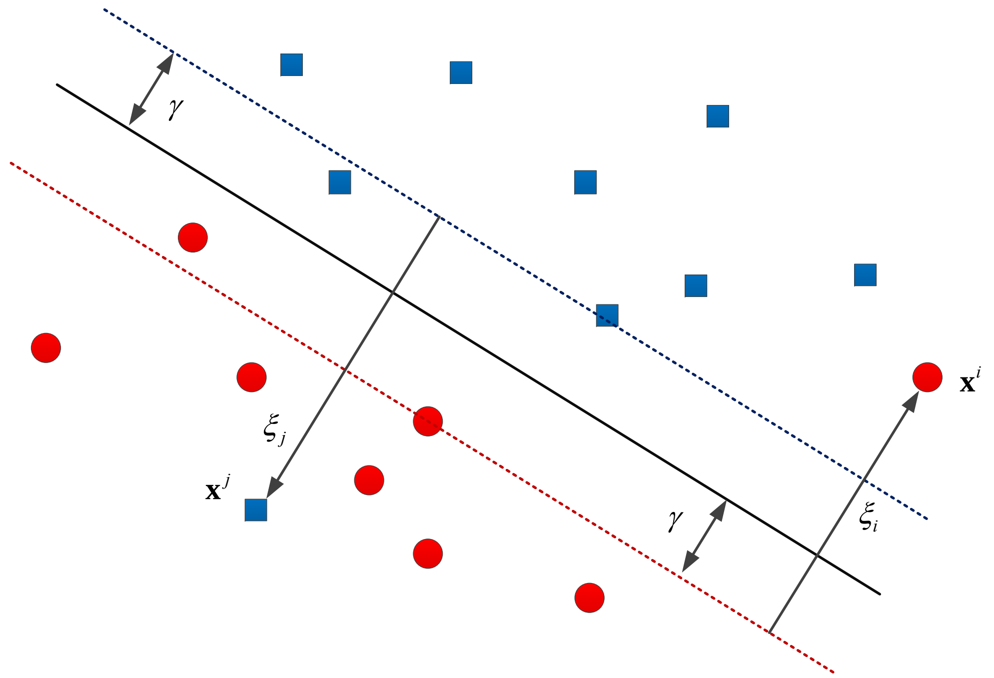
If data are not linearly separable:

Primal problem is infeasible
Dual problem is unbounded (Could it also infeasible?)

In 1995, Cortes and Vapnik suggested a modified maximum margin idea that allows for outliers or mislabeled examples. The proposed method is called Soft Margin method, which choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split data samples.

6.1. Origins of SVM

6.1.2 Robust Linear Discrimination



We can introduce the slack variable ξ_i for each training point, which measure the degree of misclassification of the datum x_i, to get a robust linear discrimination.

6.1. Origins of SVM

6.1.2 Robust Linear Discrimination

The objective function is then increased by a function which penalizes non-zero ξ_i, and the optimization becomes a trade off between a large margin, and a small error penalty.

If the penalty function is linear, the optimization problem is

$$\min \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \tag{6.19}$$

subject to $y_i(wx_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$.

6.1. Origins of SVM

6.1.2 Robust Linear Discrimination

The Lagrangian unconstrained form and the saddle point optimality conditions are

$$L(w,b,\xi,\Gamma)=\frac{1}{2}\|w\|^2+C\sum_{i=1}^{l}\xi_i-\sum_{i=1}^{l}\alpha_i[y_i(wx_i+b)-1+\xi_i]-\sum_{i=1}^{l}\delta_i\xi_i \tag{6.20}$$

$$\begin{cases}\dfrac{\partial L}{\partial w}=w-\sum_{i=1}^{l}\alpha_i y_i x_i=0\\[2ex]\dfrac{\partial L}{\partial b}=\sum_{i=1}^{l}\alpha_i y_i=0\\[2ex]\dfrac{\partial L}{\partial \xi_i}=C-\alpha_i-\delta_i=0\end{cases} \tag{6.21}$$

6.1. Origins of SVM

6.1.2 Robust Linear Discrimination

The Lagrangian dual problem is then formulated as

$$\max \quad \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j x_i^T x_j \qquad (6.22)$$

s.t. $\sum_{i=1}^{l} \alpha_i y_i = 0$ and $0 \le \alpha_k \le C$ (Dual feasibility changed!).

The Karush-Kuhn-Thucker KKT condition is

$$\alpha_i \cdot [y_i((x_i \cdot w) + b) - 1 + \xi_i] = 0, \quad i = 1, \ldots, \quad l = N + M \qquad (6.23)$$

6.1. Origins of SVM

6.1.2 Robust Linear Discrimination

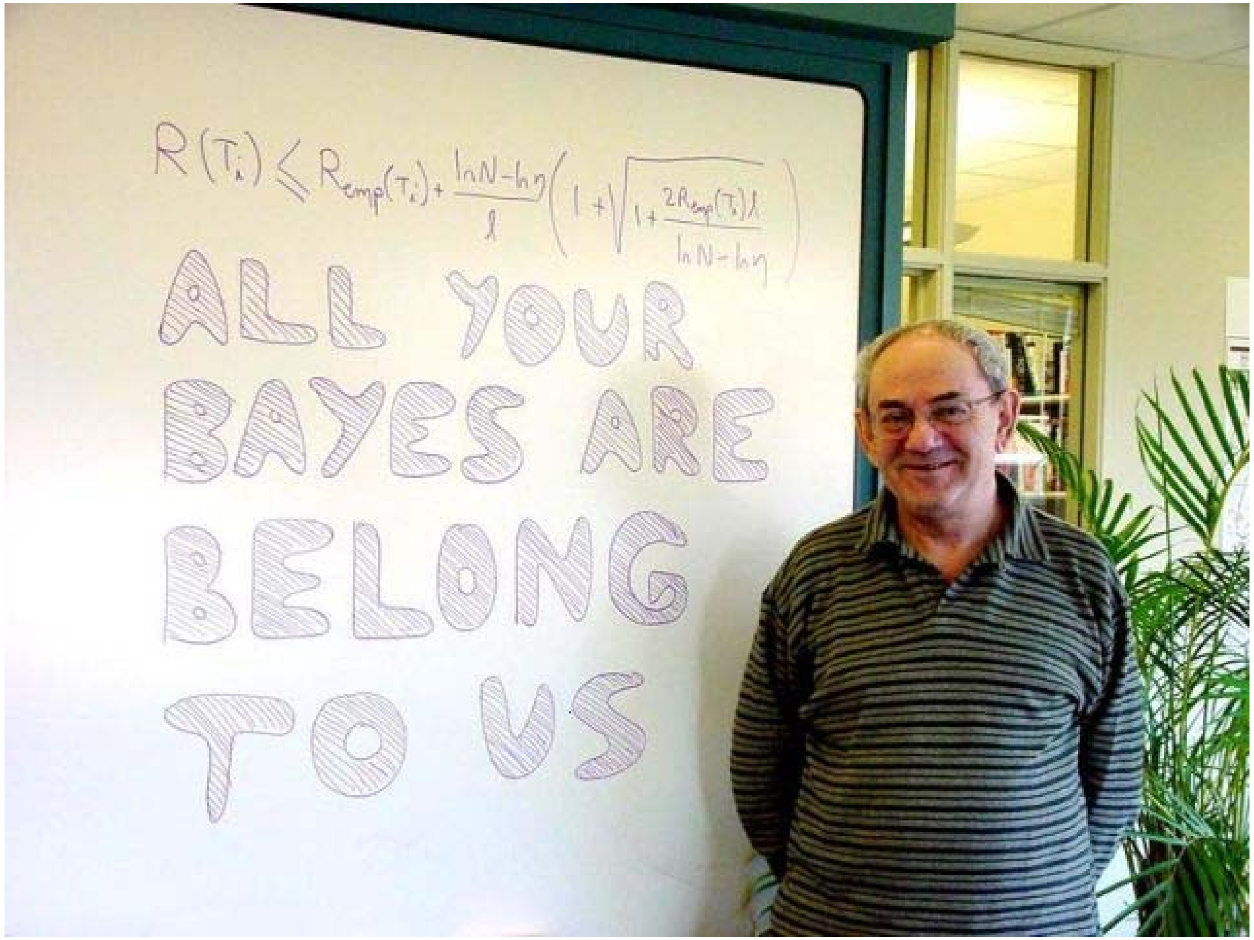
Clearly, the non-zero α_i correspond to

$$y_i((x_i \cdot w) + b) = 1 - \xi_i, \quad i = 1, \ldots, \quad l = N + M \qquad (6.24)$$

The key advantage of a linear penalty function is that the slack variables vanish from the dual problem, with the constant C appearing only as an additional constraint on the Lagrange multipliers. Cortes and Vapnik received the 2008 ACM Paris Kanellakis Award for the above formulation and its huge impact in practice.

6.1. Origins of SVM

6.1.3 Transductive SVM



Vladimir Naumovich Vapnik

6.1. Origins of SVM

6.1.2 Robust Linear Discrimination

How to choose the penalty function $G(\xi_i)$?

$$\min \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} G(\xi_i) \tag{6.25}$$

subject to $y_i(wx_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$.

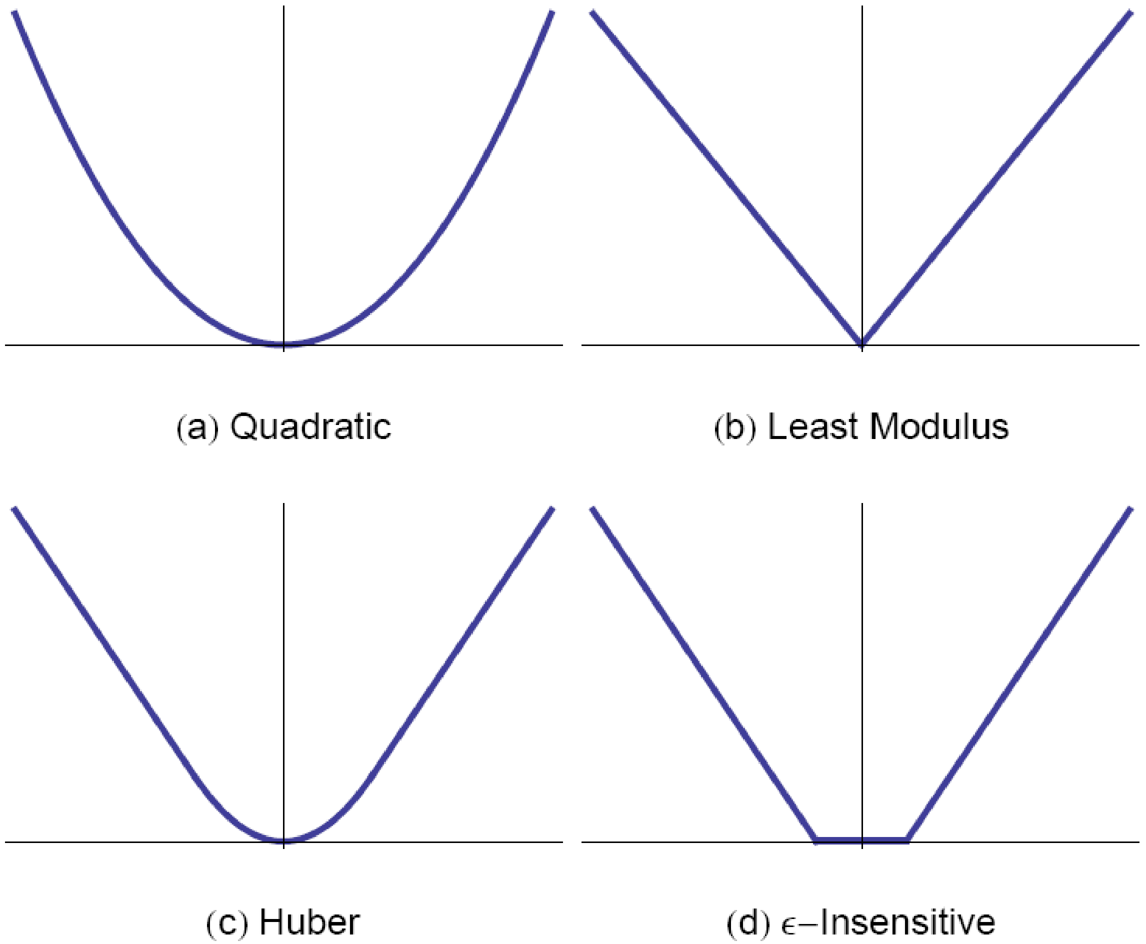
Of course, $G(\xi_i)$ must be convex!

Please tell us your preference of penaltyfunctions.

6.1. Origins of SVM

6.1.2 Robust Linear Discrimination

Some typical penalty functions

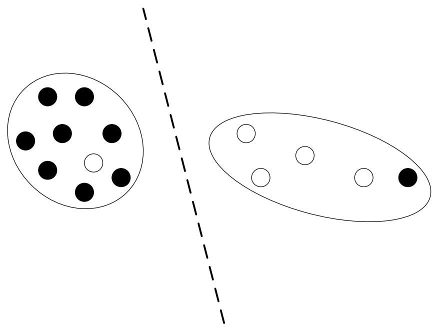


6.1. Origins of SVM

6.1.3 Transductive SVM

Inductive Inference vs Transductive Inference: inductive solution implies a transductive solution, by evaluating the decision function at the given test points, but not vice versa.

Transductive Learning vs. Semi-Supervised Learning (SSL): SSL make use of both labeled and unlabeled data for training - typically a small amount of labeled data with a large amount of unlabeled data.



6.1. Origins of SVM

6.1.3 Transductive SVM

In many real-world applications, labeling is often costly, while an enormous amount of unlabeled data is available with little cost. Transductive SVM were introduced by Vapnik and Burges to treat partially labeled data in semi-supervised learning. It considers structural properties (e.g. correlational structures) of the data set to be classified.

$$\min \frac{1}{2}\|w\|^2 \qquad (6.26)$$

subject to $y_i\left(w^T x_i + b\right) \geq 1$, $y_j\left(w^T x_j + b\right) \geq 1$, where y_i are known parameters and y_j are variables with $y_j \in \{+1,-1\}$.

6.1. Origins of SVM

6.1.3 Transductive SVM

Natural name for learning setting	Experiment name	Auxiliary data		Test data	
		Domain	$Y_{auxiliary}$	Domain	X_{test}
classic supervised learning	SuprvNonTransfer	$\mathcal{D}^{source}$	seen	$\mathcal{D}^{source}$	unseen
classic transductive learning	SuprvNonTransfer	$\mathcal{D}^{source}$	seen	$\mathcal{D}^{source}$	seen
classic transfer learning	UnsuprvTransfer	$\mathcal{D}^{source}$	seen	$\mathcal{D}^{target}$	unseen
transductive transfer learning	UnsuprvTransfer	$\mathcal{D}^{source}$	seen	$\mathcal{D}^{target}$	seen
classic semi-supervised learning		$\mathcal{D}^{source}$	unseen	$\mathcal{D}^{source}$	unseen
transductive semi-supervised learning		$\mathcal{D}^{source}$	unseen	$\mathcal{D}^{source}$	seen
semi-supervised transfer learning		$\mathcal{D}^{source}$	unseen	$\mathcal{D}^{target}$	unseen
transductive semi-supervised transfer learning		$\mathcal{D}^{source}$	unseen	$\mathcal{D}^{target}$	seen
reverse-transfer supervised learning[1]		$\mathcal{D}^{target}$	seen	$\mathcal{D}^{source}$	unseen
reverse-transfer transductive supervised learning[1]		$\mathcal{D}^{target}$	seen	$\mathcal{D}^{source}$	seen
supervised inductive transfer learning[2]	SuprvTransfer	$\mathcal{D}^{target}$	seen	$\mathcal{D}^{target}$	unseen
supervised transductive transfer learning[2]		$\mathcal{D}^{target}$	seen	$\mathcal{D}^{target}$	seen
reverse-transfer semi-supervised learning[1]		$\mathcal{D}^{target}$	unseen	$\mathcal{D}^{source}$	unseen
reverse-transfer transductive semi-supervised learning[1]		$\mathcal{D}^{target}$	unseen	$\mathcal{D}^{source}$	seen
unsupervised inductive transfer learning[2]		$\mathcal{D}^{target}$	unseen	$\mathcal{D}^{target}$	unseen
unsupervised transductive transfer learning[2]	UnsuprvTransfer	$\mathcal{D}^{target}$	unseen	$\mathcal{D}^{target}$	seen

[1] These settings and names are unusual and not likely in practice, but are included for completeness.
[2] Equivalent to its classic version if we exclude the $\mathcal{D}^{source}_{train}$ data.

6.2. More about SVM

6.2.1 Separable Problems and Kernels

From the viewpoint of convex optimization, we can see that to separate two sets of points by a nonlinear function

$$f(x_i) > 0,\ i = 1, \ldots, N,\quad f(y_i) < 0,\ i = 1, \ldots, M \tag{6.27}$$

Sometimes, we can choose a linearly parameterized family of functions $f(z) = \theta^T F(z)$, $F = (F_1, \ldots, F_k) : R^n \to R^k$ are basis functions. Then, we solve a set of linear inequalities in θ:

$$\theta^T F(x_i) \geq 1,\ i = 1, \ldots, N,\quad \theta^T F(y_i) \leq -1,\ i = 1, \ldots, M \tag{6.28}$$

6.2. More about SVM

6.2.1 Separable Problems and Kernels

quadratic discrimination: $f(z) = z^T P z + q^T z + r$

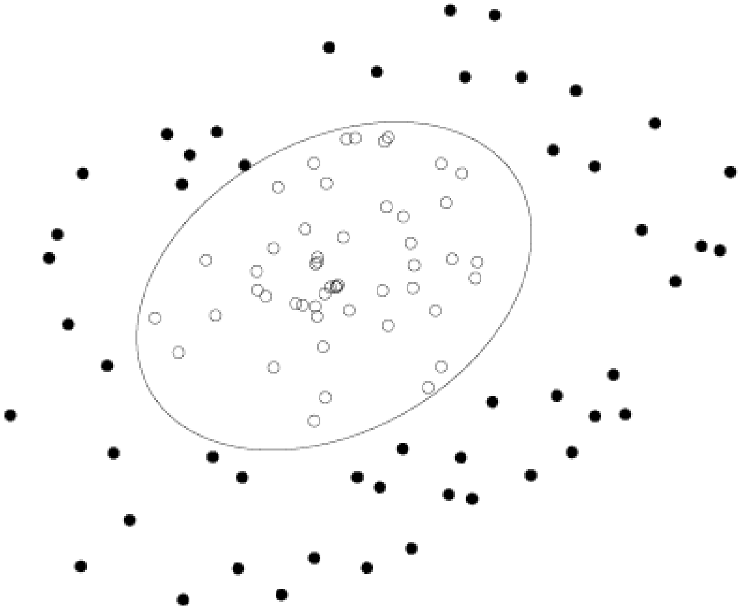
$$x_i^T P x_i + q^T x_i + r \geq 1, \qquad y_i^T P y_i + q^T y_i + r \leq -1$$

we can add additional constraints (e.g., $P \leq -I$ to separate by an ellipsoid)

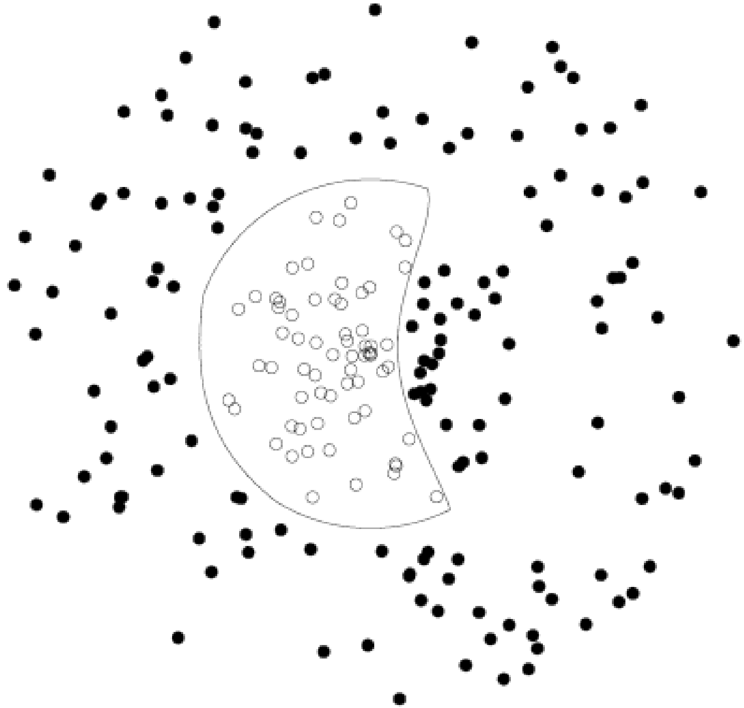
polynomial discrimination: $F(z)$ are all monomials up to a given degree

6.2. More about SVM

6.2.1 Separable Problems and Kernels



separation by ellipsoid



separation by 4th degree polynomial

Page 431 on Boyd & Vandenberghe book

6.2. More about SVM

6.2.1 Separable Problems and Kernels

The dot product $x_p^T x_q$ in (6.17) can be replaced by a kernel function $k(x_p^T x_q) = \phi(x_p)^T \phi(x_q)$, which extends the above linear discriminant SVM to a nonlinear SVM.

$$\max_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j k_{ij} \qquad (6.29)$$

Suppose the Lagrangian prime problem is

$$L(w,b,\xi,\Gamma) = \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i [y_i (w^T \phi(x_i) + b) - 1 + \xi_i] - \sum_{i=1}^{l} \delta_i \xi_i$$

(6.30)

6.2. More about SVM

6.2.1 Separable Problems and Kernels

$$\begin{cases} \dfrac{\partial L}{\partial w} = w - \sum_{i=1}^{l} \alpha_i y_i \phi(x_i) = 0 \\ \dfrac{\partial L}{\partial b} = \sum_{i=1}^{l} \alpha_i y_i = 0 \\ \dfrac{\partial L}{\partial \xi_i} = C - \alpha_i - \delta_i = 0 \end{cases} \qquad (6.31)$$

The Karush-Kuhn-Thucker KKT condition is

$$\alpha_i \cdot [y_i(w^T \phi(x_i) + b) - 1 + \xi_i] = 0, \quad i = 1, \ldots, \quad l = N + M \qquad (6.32)$$

6.2. More about SVM

6.2.1 Separable Problems and Kernels

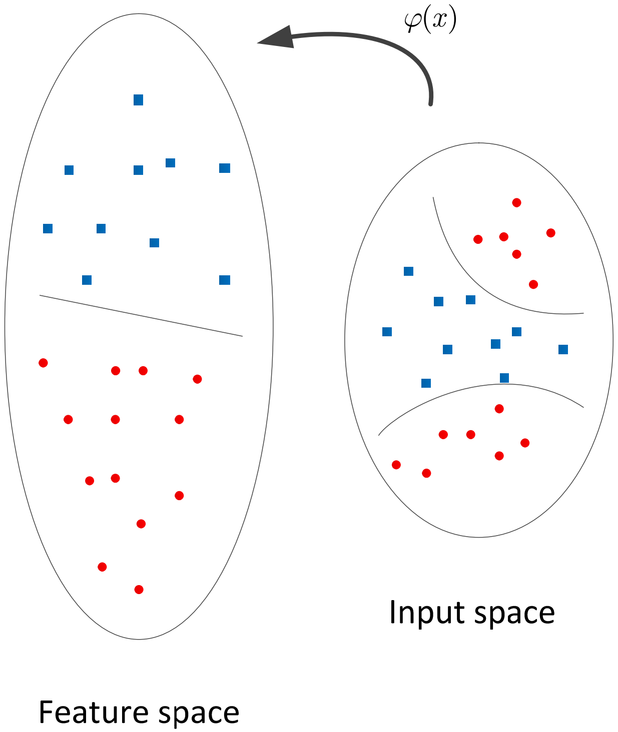
$$\delta_i \xi_i = 0, \quad i = 1, \ldots, \quad l = N + M \tag{6.33}$$

If we only know $k(x_p^T x_q)$ but do not know $\phi(x_q)$, the new decision function is

$$\begin{aligned} f(x, \hat{w}, b) &= \mathrm{sgn}(\hat{w}^T \phi(x) + b) \\ &= \mathrm{sgn}\left(\sum_{k=1}^{l} \alpha_i y_i \phi(x_i)^T \phi(x) + b \right) \\ &= \mathrm{sgn}\left(\sum_{k=1}^{l} \alpha_i y_i k(x, x_i) + b \right) \end{aligned} \tag{6.34}$$

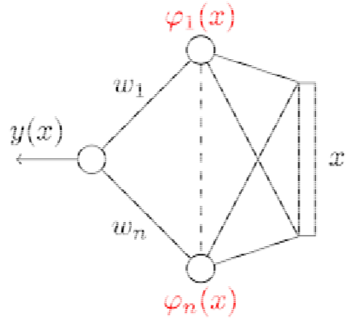
6.2. More about SVM

6.2.1 Separable Problems and Kernels



Primal space: (→large data sets)

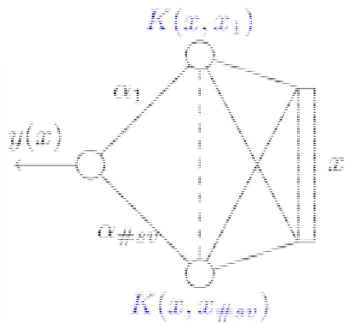
Parametric: estimate $w \in \mathbb{R}^n$
$y(x) = \text{sign}[w^T \varphi(x) + b]$



$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$ ("Kernel trick")

Dual space: (→high dimensional inputs)

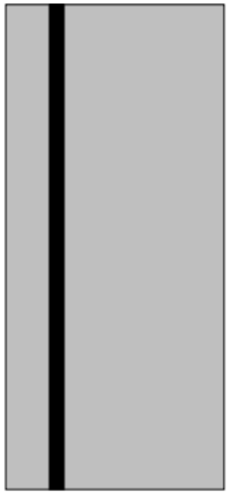
Non-parametric: estimate $\alpha \in \mathbb{R}^N$
$y(x) = \text{sign}[\sum_{i=1}^{\#sv} \alpha_i y_i K(x, x_i) + b]$



6.2. More about SVM

6.2.1 Separable Problems and Kernels

Example 1: microarray data (10.000 genes & 50 training data)



Classifier model:

$\text{sign}(w^T x + b)$ (primal)

$\text{sign}(\sum_i \alpha_i y_i x_i^T x + b)$ (dual)

primal: $w \in \mathbb{R}^{10.000}$ (only 50 training data!)

dual: $\alpha \in \mathbb{R}^{50}$

Example 2: datamining problem (1.000.000 training data & 20 inputs)

primal: $w \in \mathbb{R}^{20}$

dual: $\alpha \in \mathbb{R}^{1.000.000}$ (kernel matrix: $1.000.000 \times 1.000.000$!)

6.2. More about SVM

6.2.1 Separable Problems and Kernels

We can set up the feature Space, where the input space are mapped to some other dot product space (feature space) via a nonlinear mapping, where the data points are separable.

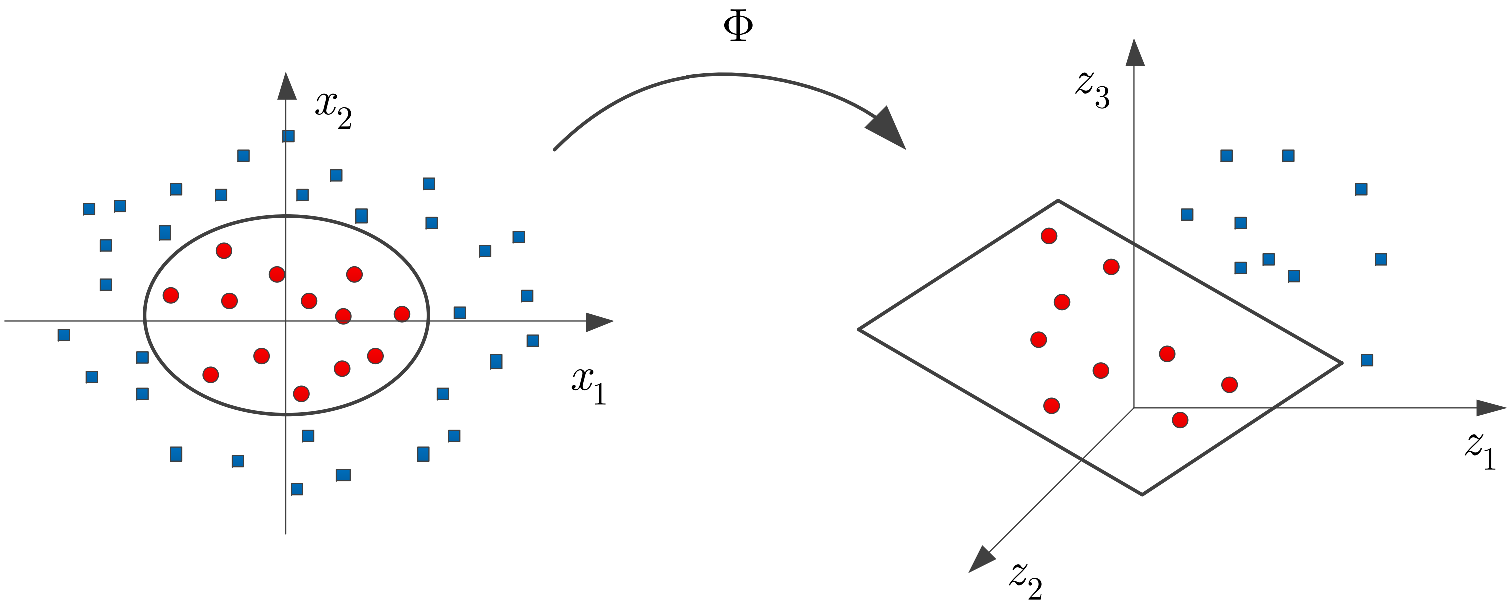
$$\Phi = I^P \rightarrow F^Q \tag{6.35}$$

Such mapping are usually viewed as functions of Kernels.

Dot products can be evaluated by some simple kernels, e.g. polynomial kernels $k(x, y) = (x \cdot y)^d$. For example, $d = 2$

6.2. More about SVM

6.2.1 Separable Problems and Kernels



$$\Phi : (x_1, x_2) \rightarrow (x_1{}^2, \sqrt{2}x_1x_2, x_2{}^2)$$

$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1 \rightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$

6.2. More about SVM

6.2.1 Separable Problems and Kernels

$$(x \cdot y)^2 = \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \right)^2 = \left(\begin{bmatrix} x_1^2 \\ \sqrt{2}\, x_1 x_2 \\ x_2^2 \end{bmatrix} \cdot \begin{bmatrix} y_1^2 \\ \sqrt{2}\, y_1 y_2 \\ y_2^2 \end{bmatrix} \right)$$

$$= (\Phi(x) \cdot \Phi(y)) = K(x, y)$$

The kernel can be inner product in infinite dimensional space.

For example, assume $x \in R^1$ and $\gamma > 0$.

$$\phi(x) = e^{-\gamma x^2} \left[1, \sqrt{\frac{2\gamma}{1!}} x, \sqrt{\frac{(2\gamma)^2}{2!}} x^2, \sqrt{\frac{(2\gamma)^3}{3!}} x^2, \cdots \right]^T$$

6.2. More about SVM

6.2.1 Separable Problems and Kernels

$$e^{-\gamma\|x_i-x_j\|^2} = e^{-\gamma(x_i-x_j)^2} = e^{-\gamma x_i^2+2\gamma x_i x_j-\gamma x_j^2}$$

$$= e^{-\gamma x_i^2-\gamma x_j^2}\left(1+\frac{2\gamma x_i x_j}{1!}+\frac{(2\gamma x_i x_j)^2}{2!}+\frac{(2\gamma x_i x_j)^3}{3!}+\cdots\right)$$

$$= e^{-\gamma x_i^2-\gamma x_j^2}\left(1\cdot 1+\sqrt{\frac{2\gamma}{1!}}x_i\cdot\sqrt{\frac{2\gamma}{1!}}x_j+\sqrt{\frac{(2\gamma)^2}{2!}}x_i\cdot\sqrt{\frac{(2\gamma)^2}{2!}}x_i+\cdots\right)=\phi(x_i)^T\phi(x_j),$$

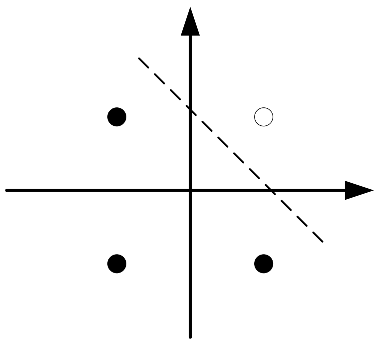
It is something like the Radial Basis Function (RBF) Neural Network, but the ideas behind are different.

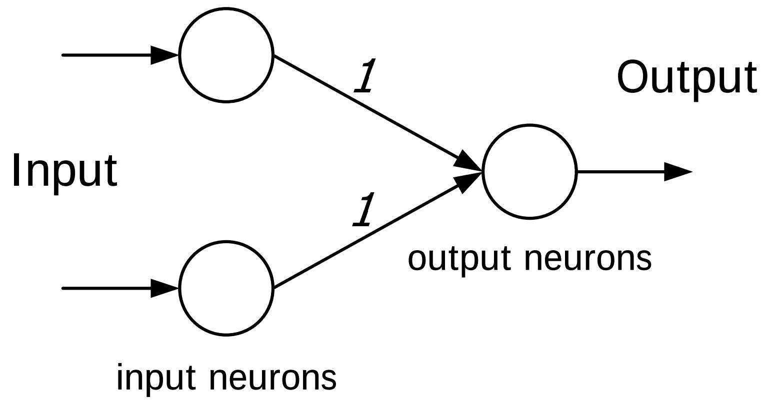
6.2. More about SVM

6.2.1 Separable Problems and Kernels

One important problem in linear discrimination: the separable case and the NON separable case

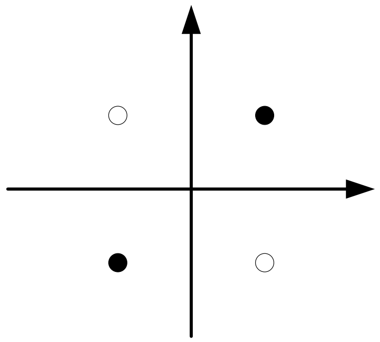
Marvin Minsky and Seymour Papert pointed out that XOR problems cannot be solved using a simple linear discrimination, but can be solved using a two layer linear discrimination

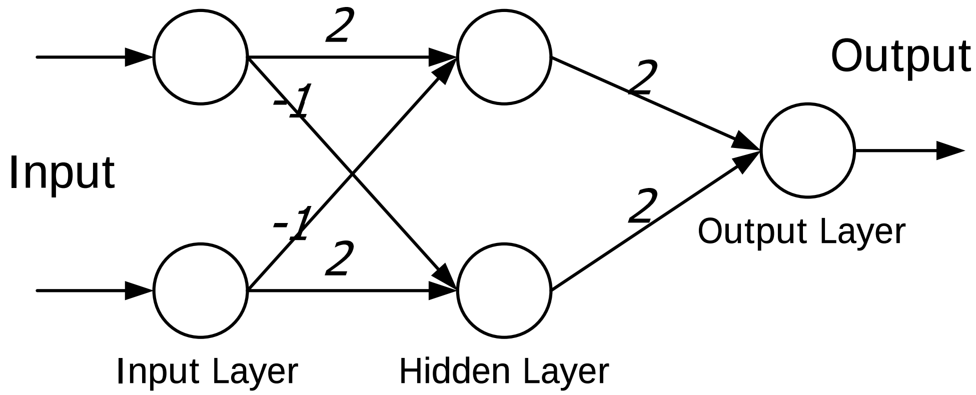




6.2. More about SVM

6.2.1 Separable Problems and Kernels





Artificial Neural Networks (ANN) is from biology to machine learning; while Support Vector Machine (SVM) is from math to machine learning

ANN uses multi-layer perceptrons to overcome the separable problems; while SVM uses the kernel tricks

6.2. More about SVM

6.2.1 Separable Problems and Kernels

Selection of kernels is usually done by cross-validations and comparision.

The follows are the most frequently used kernels:

Linear kernel, $k_{pq} = x_p^T x_q$

polynomial kernel, $k_{pq} = x_p^T P x_q$ or $k_{pq} = x_p^T P x_q + I$

RBF kernel, e.g. Gaussian kernel,

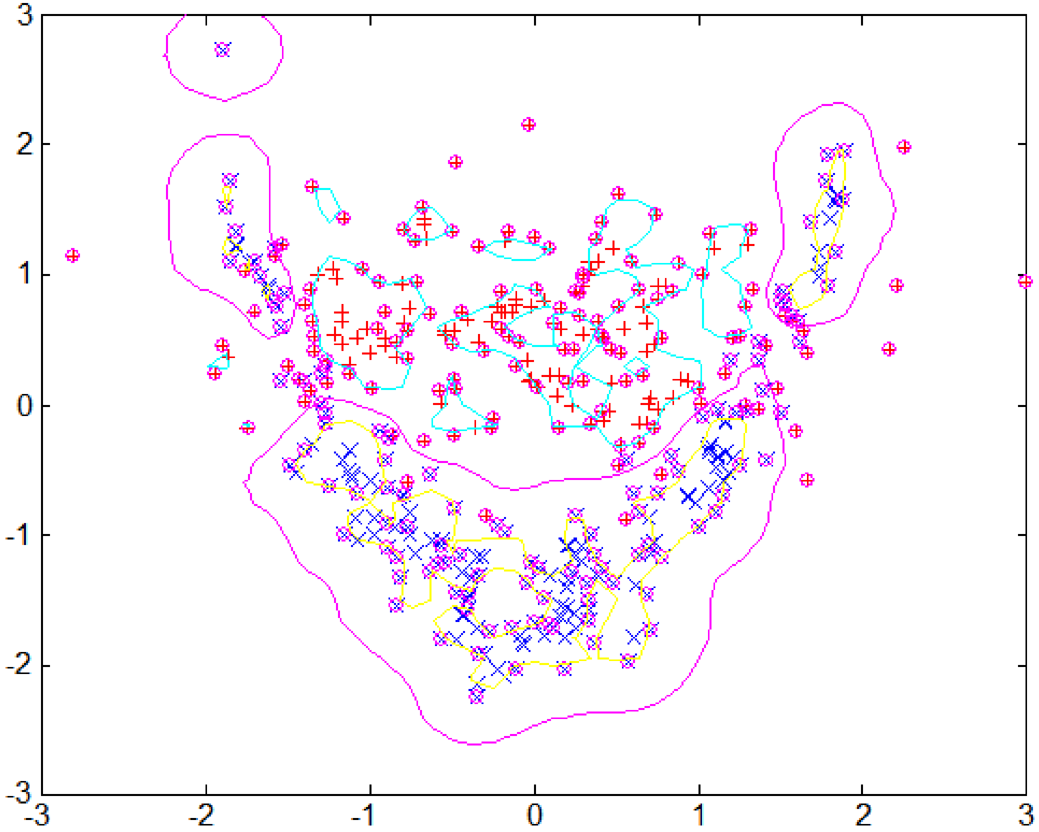
$$k_{pq} = \exp\left[-\left(x_p - x_q\right)^T P x\left(x_p - x_q\right)\right]$$

sigmoid kernel, may not be positive definite. What should we do? Lin, Lin [51]

6.2. More about SVM

6.2.1 Separable Problems and Kernels

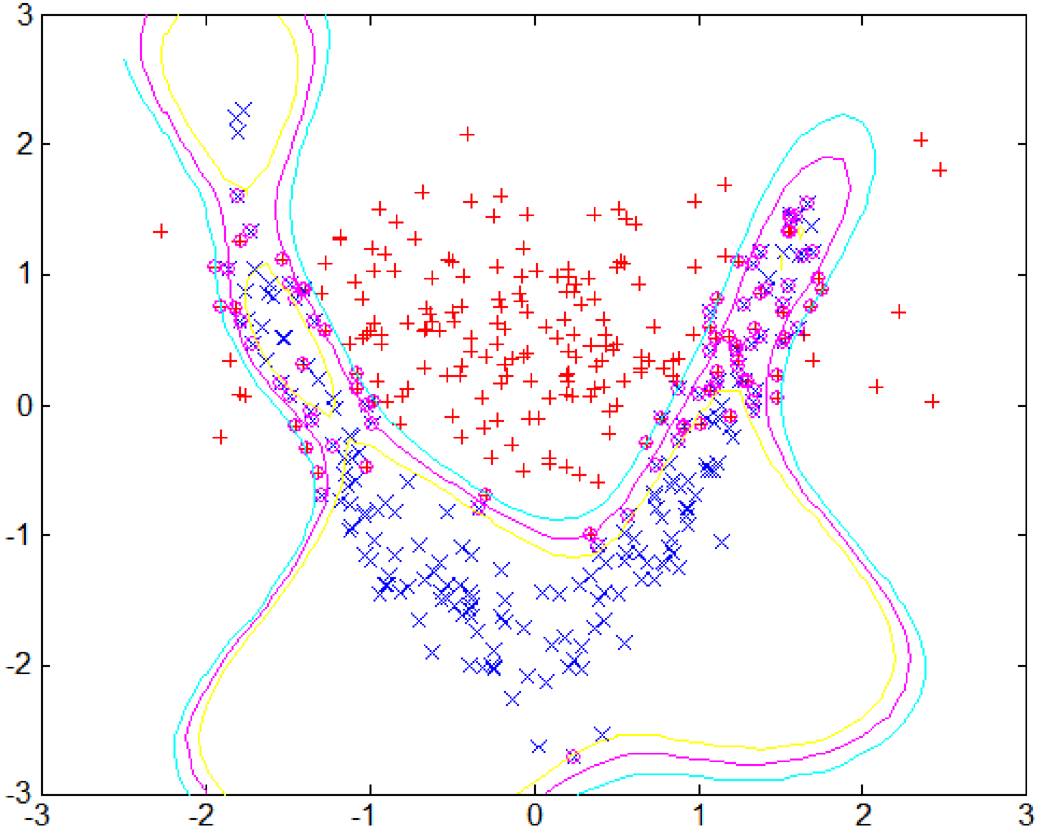
Gaussian kernel: Small bandwidth and large C



6.2. More about SVM

6.2.1 Separable Problems and Kernels

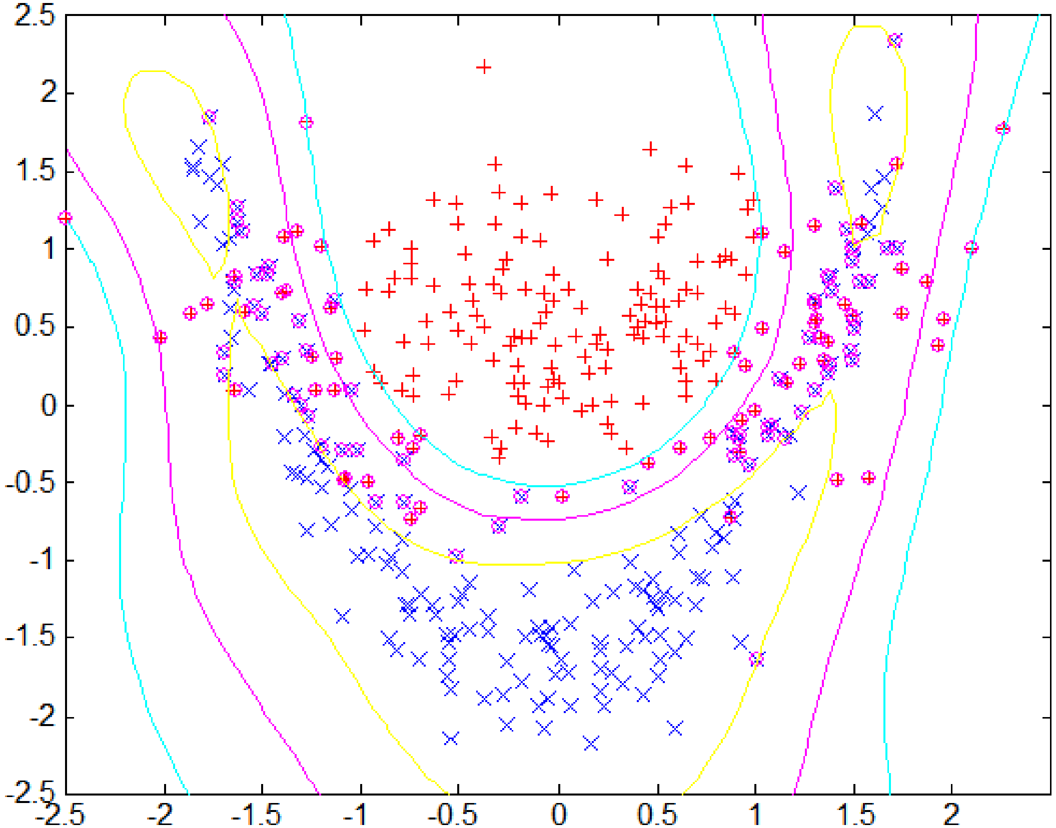
Gaussian kernel: large bandwidth and large C



6.2. More about SVM

6.2.1 Separable Problems and Kernels

Gaussian kernel: large bandwidth and small C



6.2. More about SVM

6.2.2 Training Issues: Direct Approach

The gradient ascent method based solution

$$\begin{cases} \dfrac{\partial W(\alpha)}{\partial \alpha_i} = 1 - y_i \sum_{j=1}^{l} \alpha_j y_j K(x_i, x_j) \\ \alpha_i \leftarrow \alpha_i + \eta \dfrac{\partial W(\alpha)}{\partial \alpha_i} \\ \qquad \leftarrow \quad \alpha_i + \dfrac{\omega}{K(x_i, x_i)} \left(1 - y_i \sum_{j=1}^{l} \alpha_j y_j K(x_i, x_j) \right) \end{cases} \tag{6.36}$$

where η is the learning rate. Usually $\omega \in (0,2)$. Stochastic gradient ascent gives excellent approximation in most cases.

6.2. More about SVM

6.2.2 Training Issues: Direct Approach

Handle the bound of α_i as

$$\begin{cases} \alpha_i < 0, \ \ then \ \ \alpha_i \leftarrow 0 \\ \alpha_i > C, \ \ then \ \ \alpha_i \leftarrow C \end{cases}$$

And we can stop if

$$y_i f_i(x_i) \begin{cases} \geq 1 & \alpha_i = 0 \\ = 1 & 0 < \alpha_i < C \\ \leq 1 & \alpha_i = C \end{cases}$$

6.2. More about SVM

6.2.3 Training Issues: Decomposition

Further write (6.29) as

$$\max_{\alpha} \sum_{p=1}^{l} \alpha_p - \frac{1}{2}\alpha^T Q\alpha = \max_{\alpha} 1_p^T \alpha - \frac{1}{2}\alpha^T Q\alpha \qquad (6.37)$$

subject to $\sum_{i=1}^{l} \alpha_i y_i = 0$ and $0 \le \alpha_k \le C$. If we need a closed form of Q, 50000 variables requires 10GB RAM to store Q.

$$Q_{ij} = y_i y_j k_{ij}(x_p^T x_q) = y_i y_j \phi(x_i)^T \phi(x_j) \qquad (6.38)$$

6.2. More about SVM

1.2.3 Training Issues: Decomposition

using sophisticated algorithm to calculate only activate rows or columns, Kaufman [23]

Decompose the large scale QP problem into a series of smaller QP problems

- Chunking
- Decomposition of QP with constant size matrix Q
- Approximate matrix Q with ICF
- SMO

6.2. More about SVM

6.2.3 Training Issues: Decomposition

Chunking is a classical way of decomposition, which removes all rows and columns of the matrix Q correspond to zero α. A large QP problem then breaks down into smaller QP problems.

Given training set, select an arbitrary working set B of free variables α_i, which has a new set of fixed variables.

While KKT violated (there exists some $j \in N$, such that)

$$\alpha_j = 0, f(x_j)y_j < 1, \alpha_j = C, f(x_j)y_j > 1, 0 < \alpha_j < C, f(x_j)y_j \neq 1$$

Replace α_i in B with founded α_j and solve the new QP optimization problem on the new set B'

Return α_i in B

6.2. More about SVM

6.2.3 Training Issues: Decomposition

We can solve QPs directly using standard softwares. However, the Q matrix is sometimes denseand too large to be calculated in standard softwares. So decomposition is often applied. Partition the dataset into a working set W and the remaining points R. We can rewrite the dual problem as

$$\max_{\alpha_W \in \mathbf{R}^{|W|}, \alpha_R \in \mathbf{R}^{|R|}} \sum_{\substack{i=1 \\ i \in W}}^{l} \alpha_i + \sum_{\substack{i=1 \\ i \in R}} \alpha_i - \frac{1}{2} [\alpha_{\mathbf{W}} \quad \alpha_{\mathbf{R}}] \begin{bmatrix} Q_{WW} & Q_{WR} \\ Q_{RW} & Q_{RR} \end{bmatrix} \begin{bmatrix} \alpha_{\mathbf{W}} \\ \alpha_{\mathbf{R}} \end{bmatrix} \tag{6.39}$$

subject to $\sum_{i \in W} y_i \alpha_i + \sum_{i \in R} y_i \alpha_i = 0$, $0 \le \alpha_i \le C$, $\forall i$

6.2. More about SVM

6.2.3 Training Issues: Decomposition

Suppose we have a feasible solution α. We can get a better solution by treating $\alpha_{\mathbf{W}}$ as variable and $\alpha_{\mathbf{R}}$ as constant.

$$\max_{\alpha_W \in \mathbf{R}^{|W|}} (1 - Q_{WR^{\alpha}\mathbf{R}})\alpha_W - \frac{1}{2}\alpha_{\mathbf{W}} Q_{WW^{\alpha}\mathbf{W}} \quad (6.40)$$

subject to $\sum_{i \in W} y_i \alpha_i = -\sum_{i \in R} y_i \alpha_i$, $0 \le \alpha_i \le C$, $\forall i$

The reduced problems are fixed size, and can be solved using a standard QP code. Convergence proofs are difficult, but this approach seems to always converge to an optimum in practice.

6.2. More about SVM

6.2.3 Training Issues: Decomposition

1. Given $q << l$ and α^1 as the initial solution. $1 \leftarrow k$

2. If α^k an optimum, stop. Find a working set $B \subset \{1,\dots,l\}, |B| = q$. Define $N \equiv \{1,\dots,l\} \setminus B, \alpha_B^k$ and α_N^k

3. Solve a sub-problem:

$$
\begin{aligned}
\min \quad & \frac{1}{2}\alpha_B^T Q_{BB\alpha_B} - (1_B - Q_{BN\alpha_N^k})^T \alpha_B \\
& 0 \le (\alpha_B)_i \le C, i = 1,\dots,q, \\
& y_B^T \alpha_B = -y_N^T \alpha_N^k,
\end{aligned}
$$

4. Set α_B^{k+1} and α_N^{k+1}, $k \leftarrow k+1$ and goto Step 2.

6.2. More about SVM

6.2.3 Training Issues: Decomposition

There are many different approaches to select the working set. The basic idea is

1) examine points that are not in the working set, find points which violate the reduced optimality conditions, and add them to the working set;

2) remove points which are in the working set but are far from violating the optimality conditions;

3) keep the size of the working set.

6.2. More about SVM

6.2.3 Training Issues: Decomposition

Interior-point method: get the kernel matrix $Q:(n\times n)$, while (not converge), do

$$\Delta\lambda = -\lambda + vec\left(\frac{1}{t(C-\alpha_i)}\right) + diag\left(\frac{\lambda_i}{C-\alpha_i}\right)\Delta\alpha$$

$$\Delta\xi = -\lambda + vec\left(\frac{1}{\alpha_i}\right) + diag\left(\frac{\xi_i}{\alpha_i}\right)\Delta\alpha$$

$$\Delta\nu = \frac{y^T \sum^{-1} z + y^T\alpha}{y^T \sum^{-1} y}$$

$$\Delta\alpha = \sum^{-1}(z - y\Delta\nu)\ where \sum = Q + diag\left(\frac{\xi_i}{\alpha_i} + \frac{\lambda_i}{C-\alpha_i}\right)$$

It requires inverse on a $n\times n$ matrix, $O(n^3)$

6.2. More about SVM

6.2.3 Training Issues: Decomposition

The Cholesky factorization of a positive definite matrix A is $A = LL^T$ where L is a lower triangular matrix. An Incomplete Cholesky Factorization (ICF) of A is a sparse approximation of the Cholesky factorization, which gives a sparse lower triangular matrix K that is in some sense close to L.

$$A \approx KK^T \tag{6.41}$$

It is often used to accelerate optimization (may not converge).

ICF on kernel matrix $Q = HH^T \Rightarrow (n \times n) \approx (n \times p)(p \times n)$

6.2. More about SVM

6.2.3 Training Issues: Decomposition

Interior-point method using ICF: get the approximate kernel matrix $Q = H * H^T$, while (not converge), do

$$\Delta\lambda = -\lambda + vec\left(\frac{1}{t(C-\alpha_i)}\right) + diag\left(\frac{\lambda_i}{C-\alpha_i}\right)\Delta\alpha$$

$$\Delta\xi = -\lambda + vec\left(\frac{1}{\alpha_i}\right) + diag\left(\frac{\xi_i}{\alpha_i}\right)\Delta\alpha$$

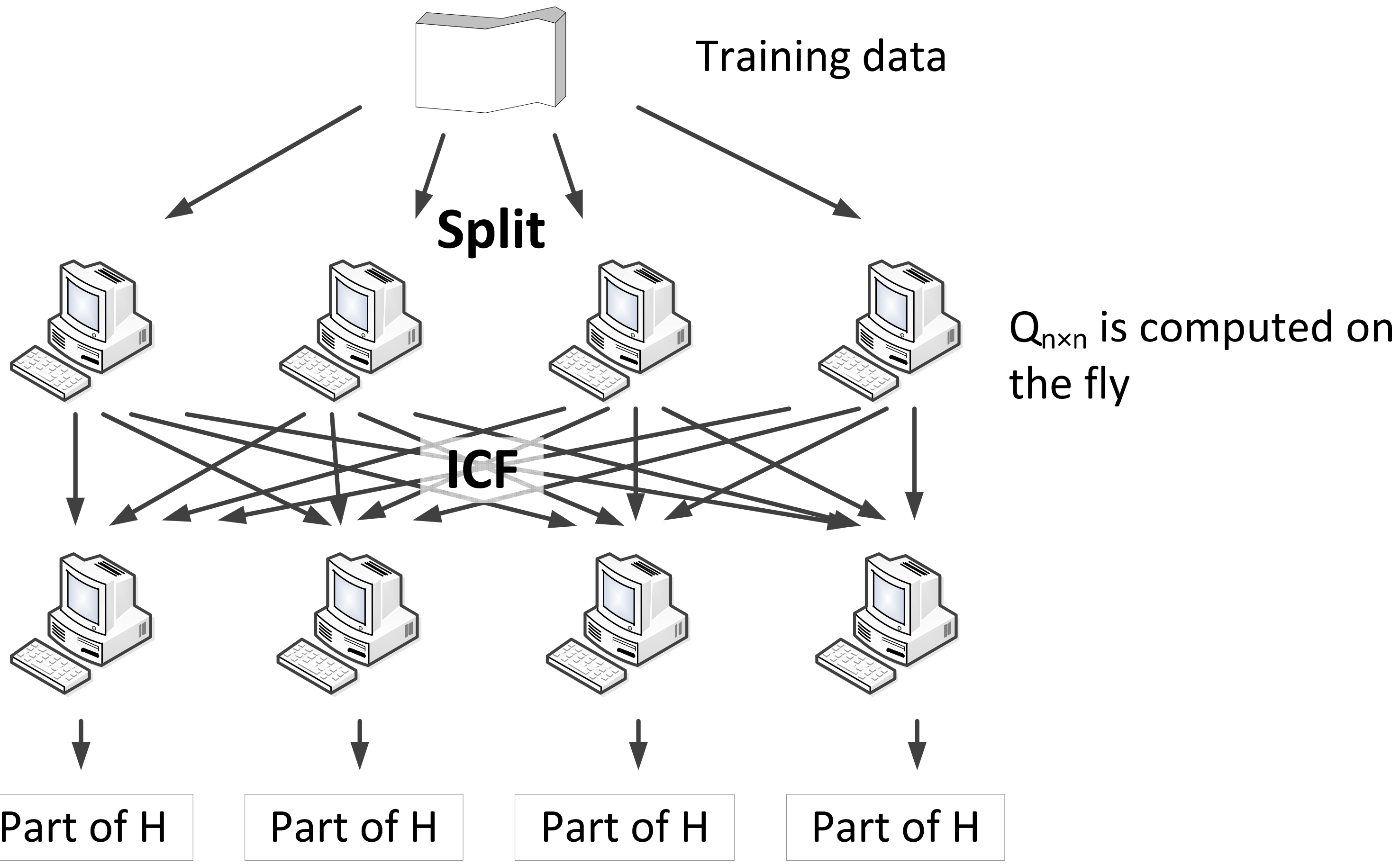
$$\Delta\nu = \frac{y^T \Sigma^{-1} z + y^T \alpha}{y^T \Sigma^{-1} y}$$

$$\Delta\alpha = \Sigma^{-1}(z - y\Delta\nu)\ where\ \Sigma^{-1} = D^{-1} - D^{-1}H\left(I + H^T D^{-1} H\right)^{-1} H^T D^{-1}$$

It requires inverse on a $p \times p$ matrix, $O(p^3)$

6.2. More about SVM

6.2.3 Training Issues: Decomposition



We can use parallel computing to accelerate our optimization.

6.2. More about SVM

6.2.3 Training Issues: Decomposition

Using approximate ICF to reduce SVM training cost (both memory and time), we have

1) Distribute training data and $H_{n \times p}$ matrix on distributed computers, thus more than 99% of computing is parallelizable (linear speedup)

2) The remaining part is still parallelizable, especially we parallelize key steps of IPM and achieve approximate linear speedup on several hundred computers

3) Can find computing locality, data can be parallel stored

6.2. More about SVM

6.2.4 Training Issues: Sequential Minimal Optimization

Sequential Minimal Optimization (SMO) is a similar method. but in each iteration, SMO chooses only two α and find their optimal value, updates the SVM to reflect new optimal value

- without any extra matrix storage.
- avoid using numerical QP optimization step each step since only two components modified. We have analytic method to solve for two lagrange multiplier.
- need more iterations to converge. But we can carry out a few operations at each step to speed-up. Convergency proof is a little bit complex.
- there are some heuristic rules for choosing α.

6.2. More about SVM

6.2.3 Training Issues: Decomposition

Because complementary slackness, we have

$$\alpha_1 y_1 + \alpha_2 y_2 = -\sum_{i=3}^{n} \alpha_i y_i \tag{6.42}$$

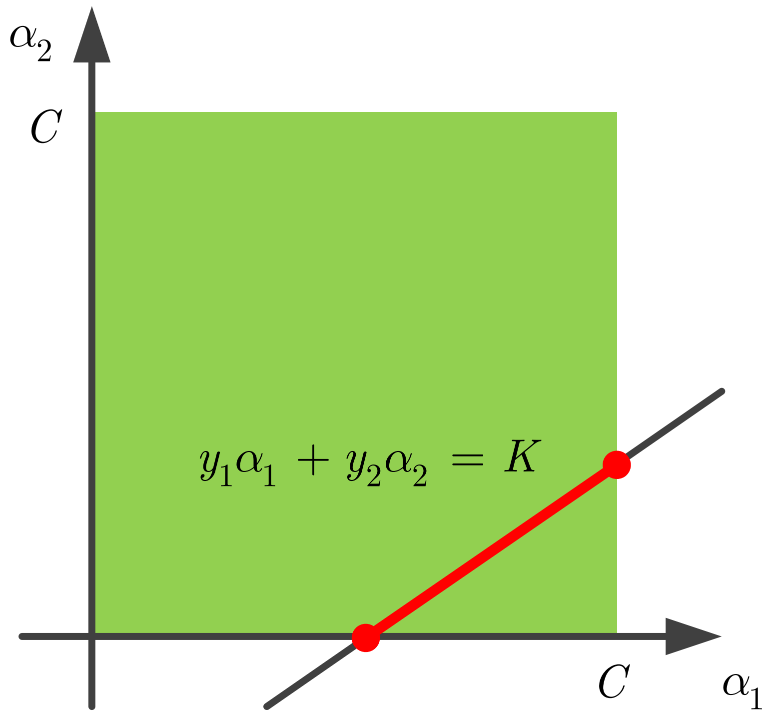
$$\alpha_2 = \frac{1}{y_2}\left(-\sum_{i=3}^{n} \alpha_i y_i - \alpha_1 y_1 \right) \tag{6.43}$$

Eliminating α_2 in the objective function, we get a quadratic optimization problem with only one variable α_1.

6.2. More about SVM

6.2.3 Training Issues: Decomposition

The only possible solution space of α_1, α_2, is



6.2. More about SVM

6.2.5 Multil-Classes Problems

Suppose we need to divide n points t_1, …, t_n ($t_i \in R^m$) into k classes. Let $x_i = (t_i, 1)$, given the associated labeling value $1 \le y_i \le k$, SVM will find k projections v_1, …, v_k to guarantee that

i) (Classification Criterion) $x_i = (t_i, 1)$ has the largest projection value on v_{y_i}, $v_{y_i}^T x_i > v_{y_j}^T x_i$, where $j \ne i$

ii) (Generalization Capability) $\min \sum_{p=1}^{k} v_p^2$

6.2. More about SVM

6.2.5 Multil-Classes Problems

Let us consider the two classes classification as an example. We will prove that we need to find two projection $v_1 = (w_1, b_1)$, $v_2 = (w_2, b_2)$ to satisfy the above requirements, but indeed we have $w_1 = -w_2$, $b_1 = -b_2$.

For those t_i labeled with $y_i = 1$, we have

$$w_1^T t_i + b_1 > w_2^T t_i + b_2 \Rightarrow (w_1 - w_2)^T t_i + (b_1 - b_2) > 0 \qquad (6.44)$$

Similarly for those t_i labeled with $y_i = 2$, we have

$$(w_1 - w_2)^T t_i + (b_1 - b_2) < 0 \qquad (6.45)$$

6.2. More about SVM

6.2.5 Multil-Classes Problems

Define $w_1' = \dfrac{w_1 - w_2}{2}$, $b_1' = \dfrac{b_1 - b_2}{2}$, $w_2' = -w_1'$, $b_2' = -b_1'$, we can still guarantee that $v_{y_i}'^T x_i > v_{y_j}'^T x_i$, where $j \neq i$, $v_1' = (w_1', b_1')$ and $v_2' = (w_2', b_2')$. Moreover, we can find that

$$\sum_{p=1}^{2} w_p'^2 + b_p'^2 \leq \sum_{p=1}^{2} w_p^2 + b_p^2 \qquad (6.46)$$

Thus, the best choice of is $w_1' = w_1 = \dfrac{w_1 - w_2}{2}$, or $w_1 = -w_2$.

6.2. More about SVM

6.2.6 LS-SVM and υ-SVM

In Least Squares Support Vector Machine (LS-SVM), the objective is

$$\min \quad \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \qquad (6.47)$$

subject to $y_i\left(w^T x_i + b\right) = 1 - \xi_i$, where y_i, $y_j \in \{+1,-1\}$.

It is easy to find that this is a Quadratic Programming problem, whose dual problem is indeed a linear equation set.

Define $y = [y_1, \ldots, y_n]^T$, $e = [1, \ldots, 1]^T$, $z = [y_1 x_1, \ldots, y_n x_n]^T$, we get

6.2. More about SVM

6.2.6 LS-SVM and υ-SVM

$$\begin{bmatrix} I & 0 & 0 & -z^T \\ 0 & 0 & 0 & -y^T \\ 0 & 0 & CI & -I \\ z^T & y^T & I & 0 \end{bmatrix} \begin{bmatrix} w \\ b \\ e \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{6.48}$$

or

$$\begin{bmatrix} 0 & -y^T \\ y & z^T z + \gamma^{-1} I \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 & -y^T \\ y & c_i c_j x_i x_j + C^{-1} I \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ e \end{bmatrix} \tag{6.49}$$

6.2. More about SVM

6.2.6 LS-SVM and υ-SVM

Schölkopf et. al. 2000 proposed the υ-SVM

$$\min \quad \frac{1}{2}\|w\|^2 - \upsilon\rho + \frac{1}{l}\sum_{i=1}^{l}\xi_i \tag{6.50}$$

subject to $y_i(wx_i + b) \geq \rho - \xi_i$, $\rho \geq 0$, $\xi_i \geq 0$, $i = 1$, …, l

If this problem has feasible solution, υ sets an upper bound on the fraction of margin errors (the percentage of the wrongly labeled samples) and sets the lower bound on the percentage of supporting vectors in all vectors, too.

6.2. More about SVM

6.2.6 LS-SVM and υ-SVM

The dual problem is

$$\max \quad -\frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j x_i x_j \tag{6.51}$$

subject to $\sum_{i=1}^{l}\alpha_i y_i = 0$, $0 \le \alpha_i \le \frac{1}{l}$, $\sum_{i=1}^{l}\alpha_\mathrm{i} \ge \upsilon$, $i = 1$, …, l

which indicates that we should keep a balance between the number of supporting vectors and the samples which had been wrongly labeled.

6.2. More about SVM

6.2.7 Suppoting Vector Regression

Regression is to find and model the relationship between a dependent variable and one or more independent variables.

If we want to minimize $\sum_{i=1}^{l} \left| y_i - w^T \phi(x_i) \right|_\varepsilon$, we usually have

$$\min \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \hat{\xi}_i) \tag{6.52}$$

subject to $w^T\phi(x_i) - y_i \le \varepsilon + \xi_i$, $y_i - w^T\phi(x_i) \le \varepsilon + \hat{\xi}_i$, ξ_i, $\hat{\xi}_i \ge 0$, $i = 1$, …, l. How is it related to SVM problems?

6.2. More about SVM

6.2.8 Other Topics

1) VC dimensions and risks

2) Reproducing Kernel Hilbert Space (RKHS)

3) Other kernels besides inner product on vector spaces (Tree Kernel, String Kernel, Path Kernel in NLP and Text Mining)

4) Imbalanced Data
https://www.zhihu.com/question/372186043

5) Variations of SVM

6.2. More about SVM

6.2.8 Other Topics

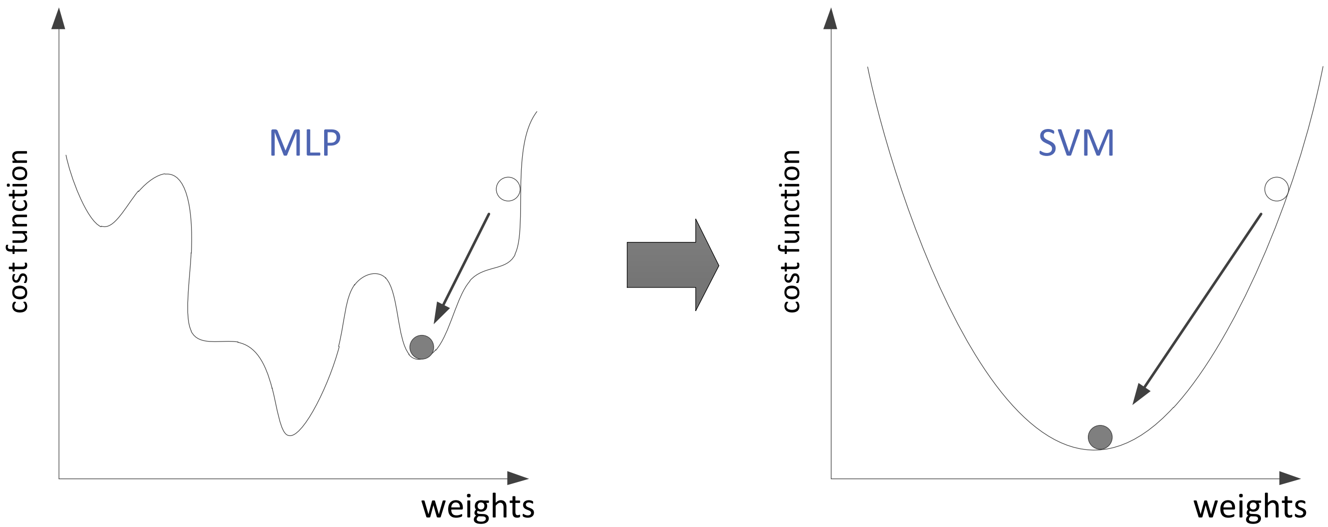
Problems of ANN

a) may exist many local minima

b) we do not know how many neurons are needed

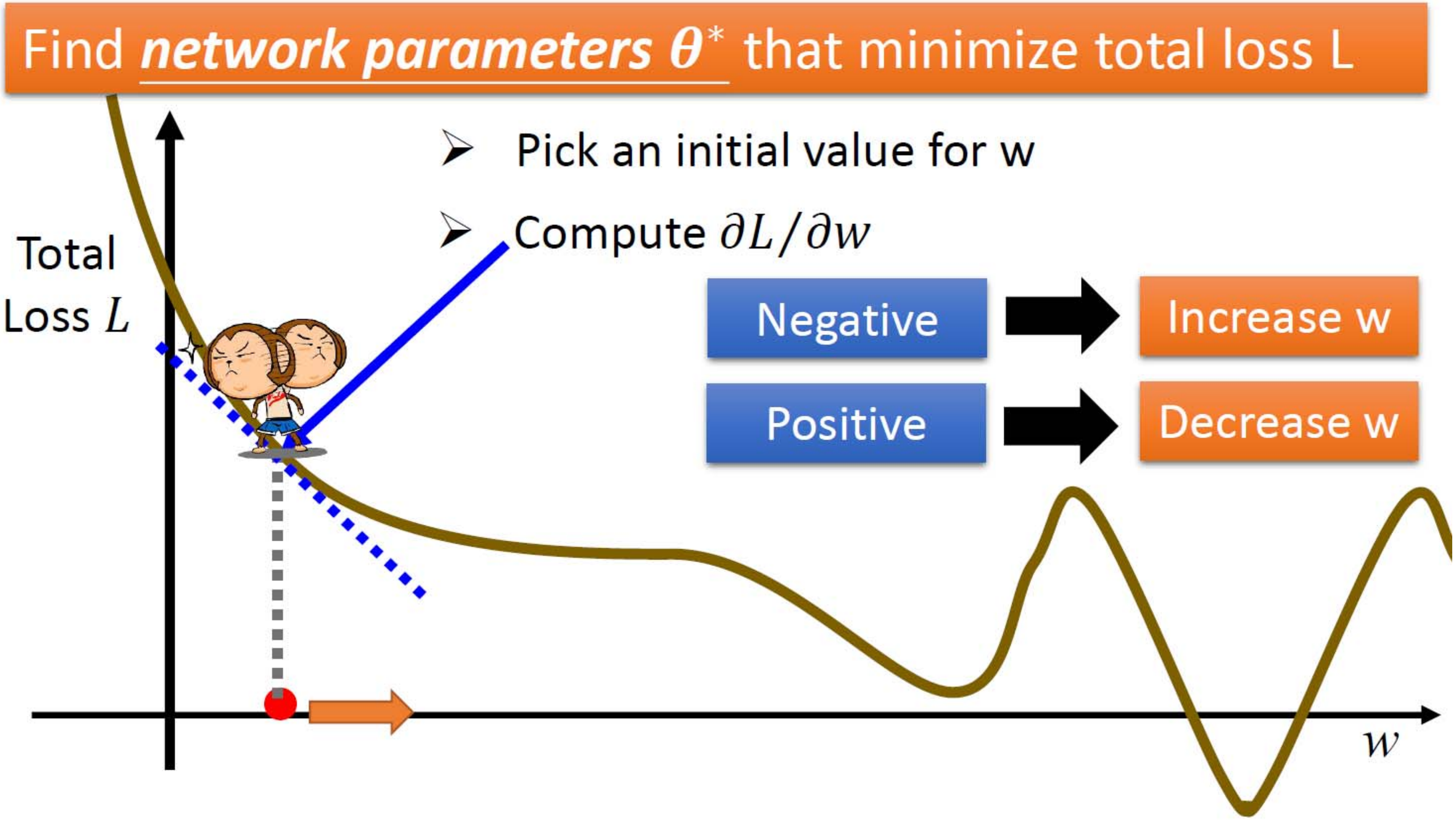
Benefits of SVM

a convex program with a unique solution



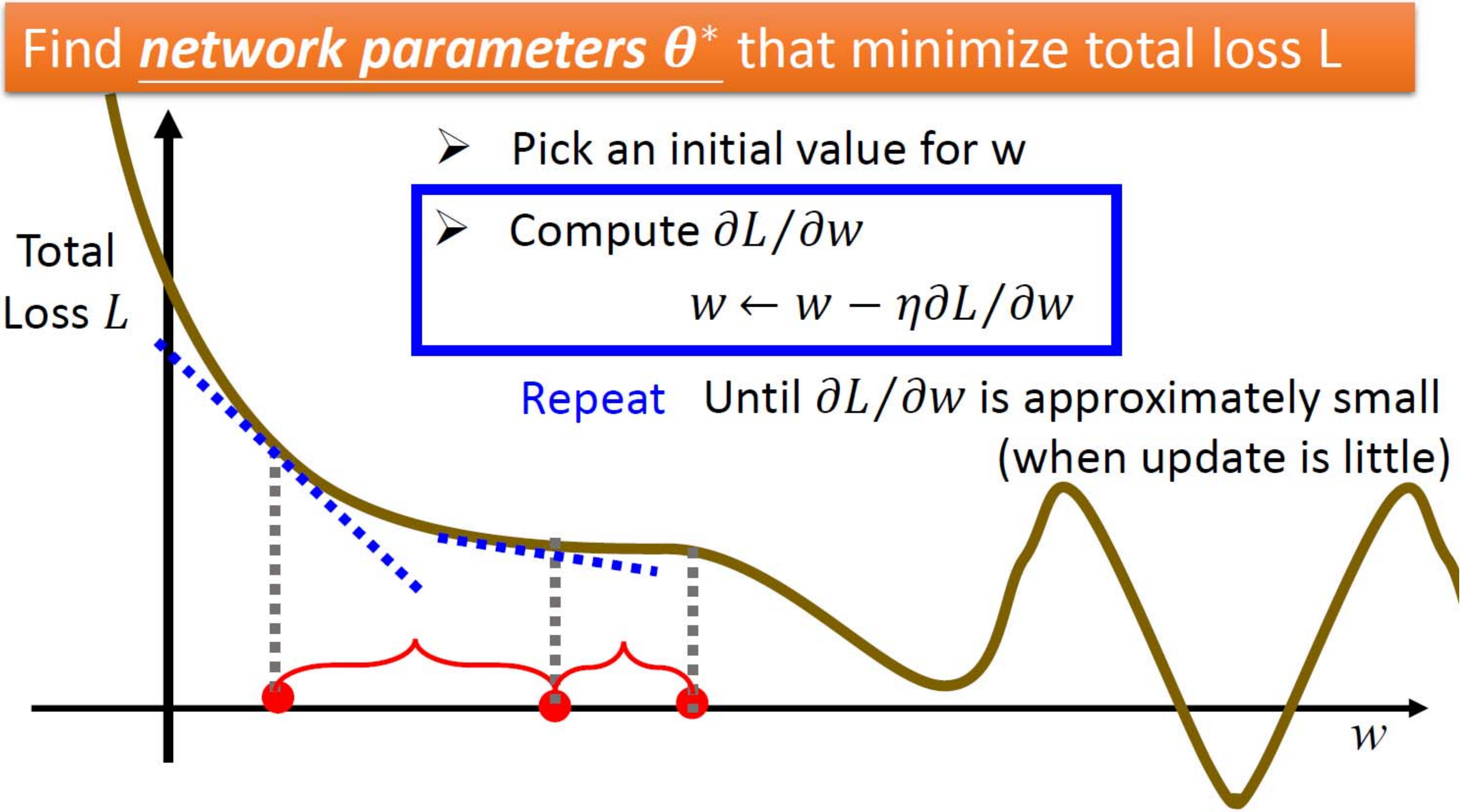
6.2. More about SVM

6.2.8 Other Topics



6.2. More about SVM

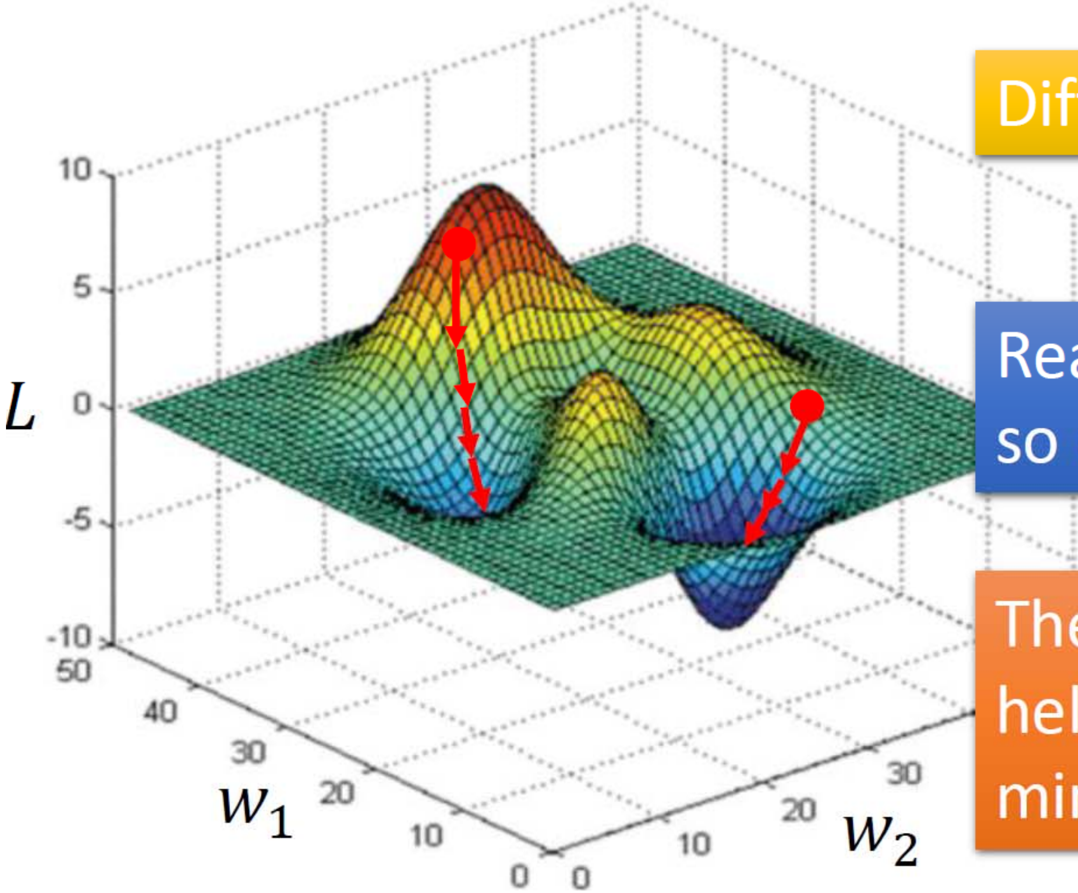
6.2.8 Other Topics



6.2. More about SVM

6.2.8 Other Topics

- Gradient descent never guarantee global minima



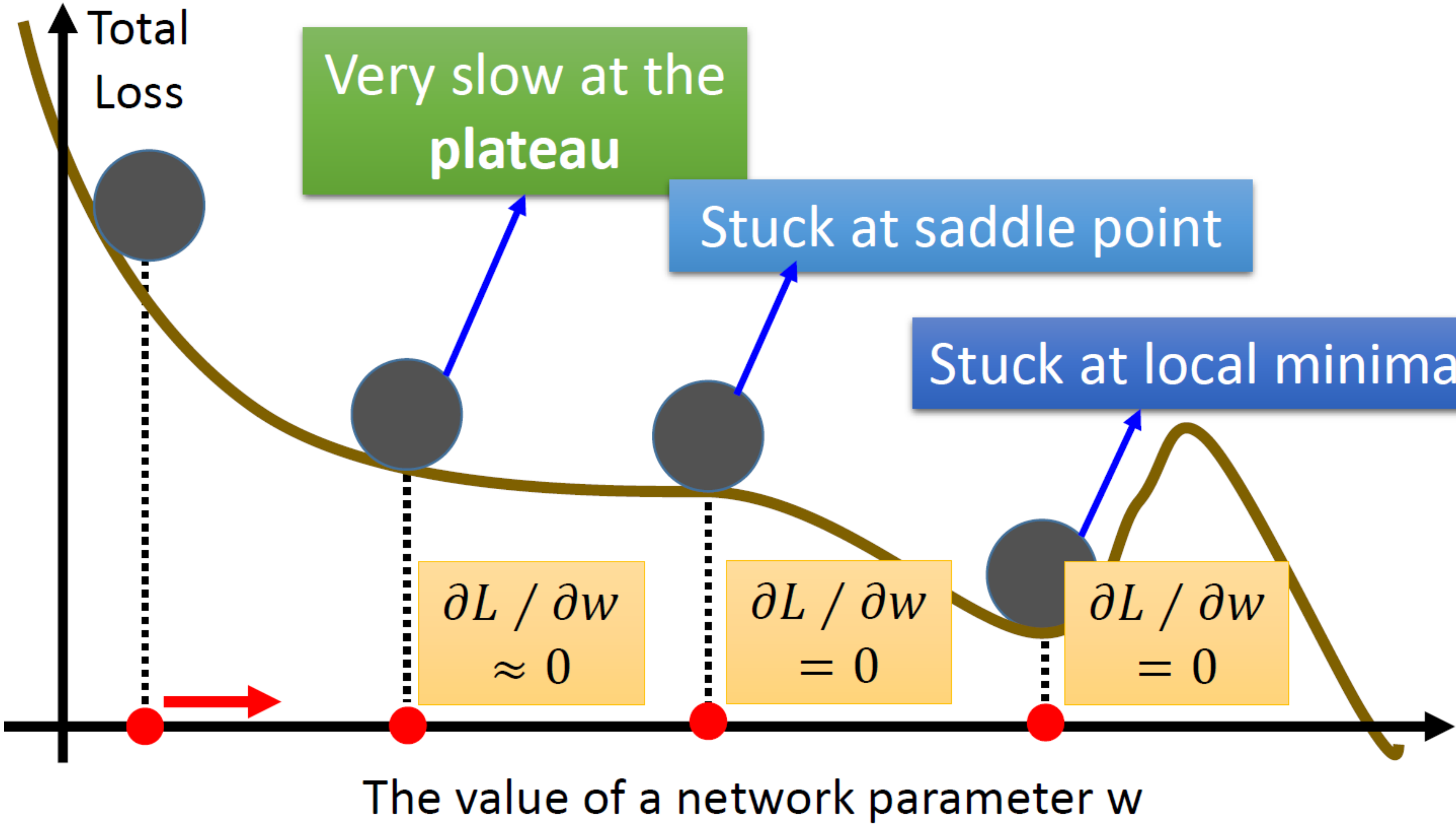
Different initial point

Reach different minima, so different results

There are some tips to help you avoid local minima, no guarantee.

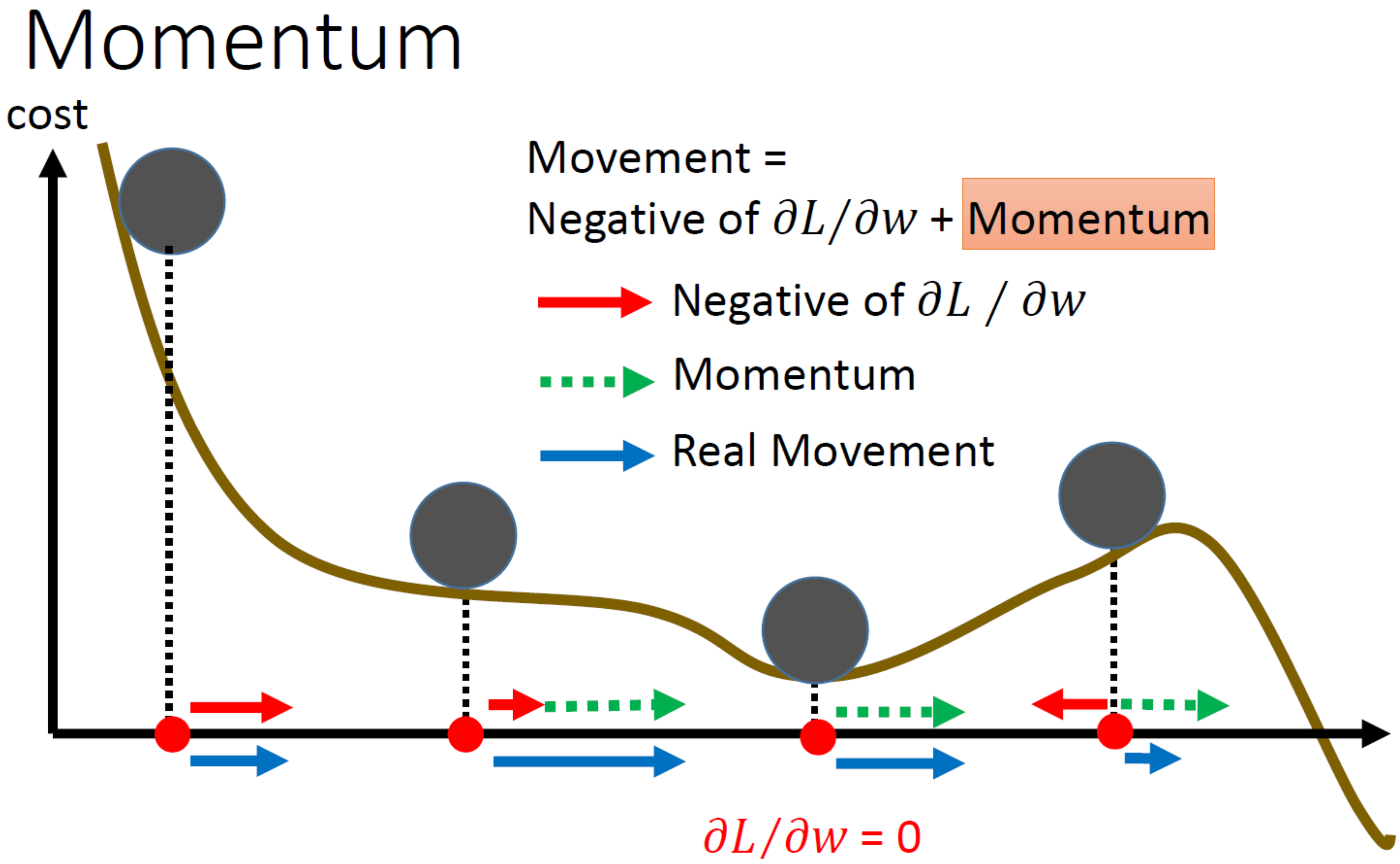
6.2. More about SVM

6.2.8 Other Topics



6.2. More about SVM

6.2.8 Other Topics



6.2. More about SVM

6.2.8 Other Topics

Do not be worried, if you do not know what we are talking about in the above.

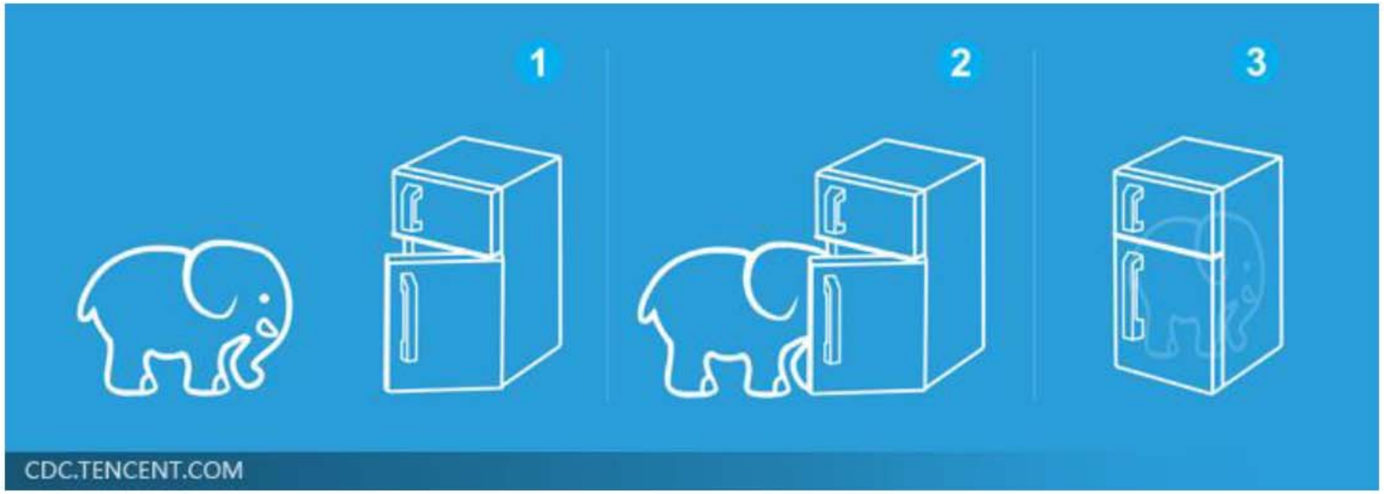
TensorFlow, MXNet, Torch, ... all these open-source software can do the difficult part of the job

At least, you should know some basic prormaming languages (e.g. C, C++, Java, Matlab, Python, ...) and get all your data ready!

6.2. More about SVM

6.2.8 Other Topics

Deep Learning is so simple ......



6.2. More about SVM

6.2.8 Other Topics

6.3. References

[1] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995.

[2] V. Vapnik, *Statistical Learning Theory*, Wiley, 1998.

[3] N. Cristianini, J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press, 2000. http://www.support-vector.net/

[4] B. Schölkopf, A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*, MIT Press, Cambridge, MA, 2002. http://www.learning-with-kernels.org/

[5] J. Shawe-Taylor, N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.

[6] L. Bottou, O. Chapelle, D. DeCoste, J. Weston, eds., *Large-Scale Kernel Machines*, The MIT Press, 2007.

[7] I. Steinwart, A. Christmann, *Support Vector Machines*, Springer-Verlag, New York, 2008.

[8] S. Abe, *Support Vector Machines for Pattern Classification*, 2nd edition, Springer, 2010.

[9] S. Sra, S. Nowozin, S. J. Wright, eds., *Optimization for Machine Learning*, MIT Press, 2012.

[10] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific, 2002.

[11] M. L. Minsky, S. A. Papert, *Perceptrons*, MIT Press, Cambridge, MA, 1969.

[12] http://en.wikipedia.org/wiki/Support_vector_machine

[13] B. Boser, I. Guyon, V. Vapnik, "A training algorithm for optimal margin classifiers," *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144-152, 1992.

[14] C. Cortes, V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995.

[15] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Knowledge Discovery and Data Mining*, vol. 2, no. 2, pp. 121-167, 1998.

[16] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, B. Schölkopf, "An introduction to kernel-based learning algorithms," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 181-201, 2001.

[17] I. Steinwart, D. Hush, C. Scovel, "Training SVMs without offset," *Journal of Machine Learning Research*, vol. 12, pp. 141-202, 2011.

[18] O. Chapelle, V. Sindhwani, S. S. Keerthi, "Optimization techniques for semi-supervised support vector machines," *Journal of Machine Learning Research*, vol. 9, pp. 203-233, 2008.

[19] O. Chapelle, *A Discussion of Semi-Supervised Learning and Transduction*, http://www.kyb.tuebingen.mpg.de/ssl-book/discussion.pdf

[20] O. Chapelle, B. Schölkopf, A. Zien, *Semi-Supervised Learning*, MIT Press, Cambridge, MA, 2006.

[21] S. Abney, *Semisupervised Learning for Computational Linguistics*, Chapman & Hall/CRC, 2008.

[22] X. Zhu, Semi-Supervised learning Literature Survey, http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf

[23] X. Zhu, A. Goldberg, *An Introduction to Semi-Supervised Learning*, Morgan & Claypool Publishers, 2009.

[24] E. Osuna, R. Freund, F. Girosi, "Training Support Vector Machines: An application to face detection," *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 130-136, 1997.

[25] L. Kaufman, "Solving the quadratic programming problem arising in support vector classification," *Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf, C. J. C. Burges, A. J. Smola, eds., MIT Press, Cambridge, MA, pp. 147-168, 1998.

[26] T. Joachims, "Making large-scale SVM learning practical," *Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf, C. J. C. Burges, A. J. Smola, eds., MIT Press, Cambridge, MA, pp. 169-184, 1998. http://www.cs.cornell.edu/People/tj/publications/joachims_98c.pdf

[27] M. Ferris, T. Munson, "Interior-point methods for massive support vector machines," *SIAM Journal on Optimization*, vol. 13, no. 3, pp. 783-804, 2002.

[28] O. Chapelle, "Training a Support Vector Machine in the Primal," *Neural Computation*, vol. 19, no. 5, pp. 1155-1178, 2007. http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/primal/

[29] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," *Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf, C. J. C. Burges, A. J. Smola, eds., MIT Press, Cambridge, MA, pp. 185-208, 1998. http://research.microsoft.com/en-us/um/people/jplatt/smotr.pdf

[30] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, K. R. K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design," *Neural Computation*, vol. 13, no. 3, pp. 637-649, 2001.

[31] S. S. Keerthi, E. G. Gilbert, "Convergence of a generalized SMO algorithm for SVM classifier design," *Machine Learning*, vol. 46, no. 1-3, pp. 351-360, 2002.

[32] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 131-159, 2002. http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/ams/

[33] K. Crammer, Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 265-292, 2001.

[34] S. S. Keerthi, S. Sundararajan, K.-W. Chang, C.-J. Hsieh, C.-J. Lin, "A sequential dual method for large scale multi-class linear SVMs," *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 408-416, 2008.

[35] R. Rifkin, A. Klautau, "In defense of one-vs-all classification," *Journal of Machine Learning Research*, vol. 5, pp. 101-141, 2004.

[36] Y. Liu, "Fisher consistency of multicategory support vector machines," *Proceedings of International Conference on Artificial Intelligence and Statistics*, 289-296, 2007.

[37] F. R. Bach, G. R. G. Lanckriet, M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," *Proceedings of ACM International Conference on Machine Learning*, 2004.

[38] S. Sonnenburg, G. Rätsch, C. Schäfer, B. Schölkopf, "Large scale multiple kernel learning," *Journal of Machine Learning Research*, vol. 7, pp. 1531-1565, 2006.

[39] M. Gönen, E. Alpaydin, "Multiple kernel learning algorithms," *Journal of Machine Learning Research*, vol. 12, pp. 2211-2268, 2011.

[40] J. A. K. Suykens, J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293-300, 1999.

[41] K. Pelckmans, J. A.K. Suykens, B. De Moor, "Morozov, Ivanov and Tikhonov regularization based LS-SVMs," *Lecture Notes in Computer Science*, vol. 3316, pp. 1216-1222, 2004.

[42] T. Evgeniou, M. Pontil, T. Poggio, "Regularization networks and support vector machines," *Advances in Computational Mathematics*, vol. 13, no. 1, pp. 1-50, 2000.

[43] R. M. Rifkin, *Everything Old is New Again a Fresh Look at Historical Approaches in Machine Learning*, Ph.D. Thesis, MIT, 2002.

[44] A. N. Tikhonov, V. A. Arsenin, *Solution of Ill-posed Problems*, Winston & Sons, Washington, 1977.

[45] M. E. Tipping, "Bayesian inference: An introduction to Principles and practice in machine learning," *Advanced Lectures on Machine Learning*, O. Bousquet, U. von Luxburg, G. Ratsch, eds., pp. 41-62. Springer, 2004. http://www.miketipping.com/papers/met-mlbayes.pdf

[46] T. Joachims, "Transductive inference for text classification using Support Vector Machines," *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 200-209, 1999.

[47] B. Schölkopf, A. J. Smola, R. C. Willamson, P. L. Barlett, "New support vector algorithms," *Neural Computation*, vol. 12, no. 5, pp. 1207-1245, 2000.

[48] D. Meyer, F. Leischa, K. Hornik, "The support vector machine under test," *Neurocomputing*, vol. 55, no. 1-2, pp. 169-186, 2003.

[49] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Transactions on Electronic Computers*, vol. 14, no. 3, pp. 326-334, 1965. http://www.stanford.edu/~cover/papers/paper2.pdf

[50] T. Gärtner, "A survey of kernels for structured data," *ACM SIGKDD Explorations Newsletter*, vol. 5, no. 1, pp. 49-58, 2003.

[51] G. R. G. Lanckriet, N. Cristianini, L. El Ghaoui, P. L. Bartlett, M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine Learning Research*, vol. 5, pp. 27-72, 2004.

[52] H.-T. Lin, C.-J. Lin, "A study on Sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods," http://www.csie.ntu.edu.tw/~cjlin/papers/tanh.pdf

[53] T. Gärtner, *Kernels for Structured Data*, World Scientific Publishing Company, 2009.

[54] M. Mohri, A. Rostamizadeh, A. Talwalkar, *Foundations of Machine Learning*, The MIT Press, 2012.

[55] C.-J. Lin, Training Support Vector Machines: Status and Challenges http://videolectures.net/icml08_lin_tsvm/
[56] A.J. Smola, B. Schölkopf, "A tutorial on support vector regression," *NeuroCOLT Technical Report*, NC-TR-98-030, Royal Holloway College, University of London, UK, 1998.
[57] LIBSVM - A Library for Support Vector Machines http://www.csie.ntu.edu.tw/~cjlin/libsvm/
[58] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871-1874, 2008.
[59] LIBLINEAR http://www.csie.ntu.edu.tw/~cjlin/liblinear/
[60] SVM Light http://www.cs.cornell.edu/People/tj/svm_light/
[61] LS-SVMlab http://www.esat.kuleuven.ac.be/sista/lssvmlab/
[62] Google pSVM http://code.google.com/p/psvm/
[63] http://www.support-vector-machines.org/SVM_soft.html
[64] http://www.support-vector-machines.org/

[65] Prof. Tomaso Poggio, Prof. Lorenzo Rosasco, Prof. Charlie Frogner, Prof. Pavan Mallapragada, *9.520: Statistical Learning Theory and Applications*, MIT, Spring 2011.

[66] Hung-yi Lee, *Deep Learning Tutorial*, 2017.