Machine Learning

Zhengchao Wan

Peking University

June 13, 2016

Overview

- Geometry of support vector machine
- Boosting

Linear classifier

Given input pattern $\boldsymbol{x} \in R^n$, consider a linear function

$$f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{x} + b \tag{1}$$

having parameters $\boldsymbol{\xi} = (\boldsymbol{w}, b)$. The machine classifies patterns into two classes C_+ and C_-, according to the signature of output function f. That is, when $f(\boldsymbol{x}, \boldsymbol{\xi}) > 0$, $\boldsymbol{x}$ is classified into C_+, and otherwise into C_-.

Linearly separable

Training set $D = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ are divided into two classes. When $\boldsymbol{x}_i \in C_+$, it's given a teacher signal $y_i = 1$, and when $\boldsymbol{x}_i \in C_-$, $y_i = -1$. They are **linearly separable** when there exists $\boldsymbol{w}$ and b, for which

$$\boldsymbol{w} \cdot \boldsymbol{x} + b > 0, \boldsymbol{x} \in C_+, \boldsymbol{w} \cdot \boldsymbol{x} + b < 0, \boldsymbol{x} \in C_- \tag{2}$$

holds. When $(\boldsymbol{w}, b)$ is such a solution, also is $(c\boldsymbol{w}, cb), \forall c > 0$. We can therefore impose the constraints

$$|\boldsymbol{w} \cdot \boldsymbol{x} + b| \geq 1, \min_i |\boldsymbol{w} \cdot \boldsymbol{x} + b| = 1. \tag{3}$$

Support vector

The Euclidean distance from point $\boldsymbol{x}$ to the separating hyperplane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ is

$$d = \frac{|\boldsymbol{w} \cdot \boldsymbol{x} + b|}{|\boldsymbol{w}|} \tag{4}$$

The minimum of distances $d_i = \frac{|\boldsymbol{w} \cdot \boldsymbol{x}_i + b|}{|\boldsymbol{w}|}$ is given by $d_{min} = \frac{1}{|\boldsymbol{w}|}$ and is attained by the points $\boldsymbol{x}_i$ that satisfy

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = 1. \tag{5}$$

We call these points the **support vectors** of the training set D and the minimal distance the **margin**.

Support vector

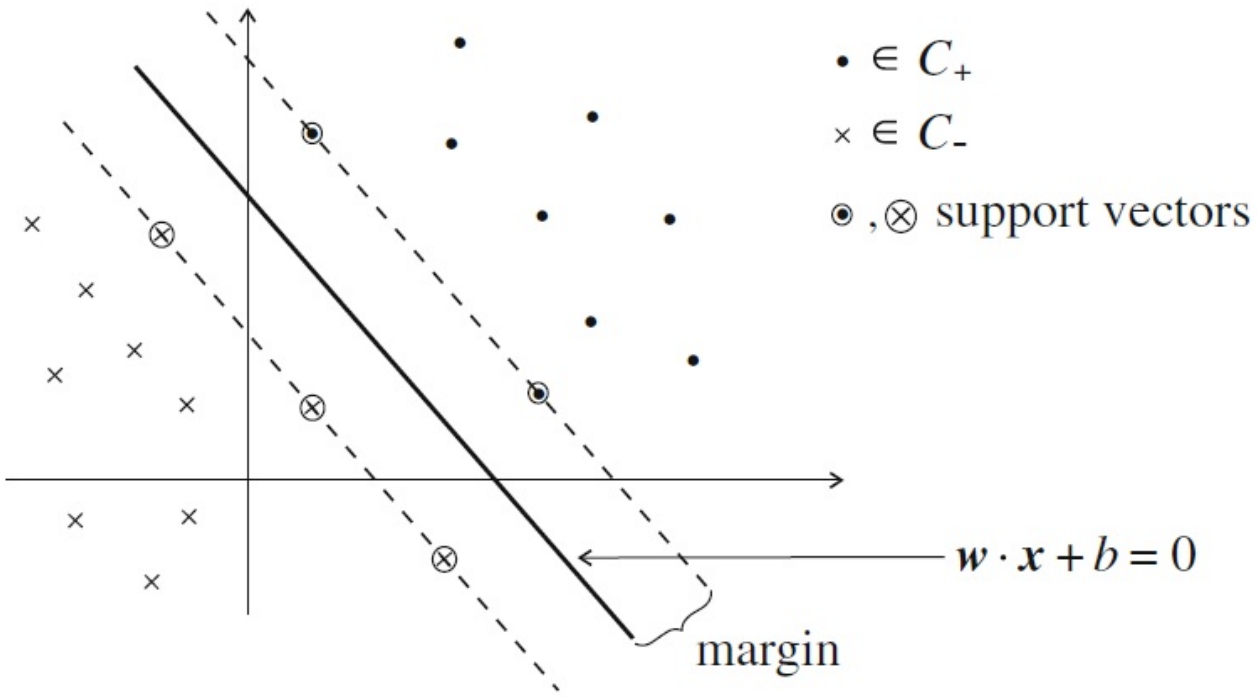

Figure: Linear classifier and support vectors

Optimal linear machine

A good machine has a large margin so we want to minimize

$$C(\boldsymbol{w}) = \frac{1}{2}|\boldsymbol{w}|^2 \tag{6}$$

under the constraint

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1. \tag{7}$$

Using Lagrange multipliers $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_N)$, we transform the problem into the unconstrained minimization of

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}|\boldsymbol{w}|^2 - \sum \alpha_i(y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - 1). \tag{8}$$

Optimal linear machine

By differentiating with $\boldsymbol{w}$ and b and making the derivatives equal to 0, we have $\sum_i \alpha_i y_i = 0$, $\boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i$ and the dual problem

$$\begin{aligned} \text{maximize } L^*(\boldsymbol{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j, \\ \alpha_i \geq 0, \sum \alpha_i y_i = 0. \end{aligned} \tag{9}$$

This is a quadratic function of α_i, which has a specific algorithm to solve. It should be remarked that $\alpha_i = 0$ when $\boldsymbol{x}_i$ is not a support vector.
The optimized output function is written as

$$f(\boldsymbol{x}, \boldsymbol{w}) = \sum \alpha_i y_i \boldsymbol{x}_i \cdot \boldsymbol{x} + b \tag{10}$$

in terms of the solution α_i, which only uses the support vectors.

Embedding into high-dimensional space

Consider a nonlinear transformation of $\boldsymbol{x} \in R^n$ into a high-dimensional space $R^m (m > n)$ by

$$\boldsymbol{z} = \boldsymbol{\varphi}(\boldsymbol{x}) = [\varphi_1(\boldsymbol{x}), \cdots, \varphi_m(\boldsymbol{x})]. \tag{11}$$

The original classification problem is transformed into a linear classification problem in R^m, where

$$f(\boldsymbol{x}, \boldsymbol{\xi}) = \boldsymbol{w} \cdot \boldsymbol{\varphi}(\boldsymbol{x}) + b, \quad \boldsymbol{\xi} = (\boldsymbol{w}, b). \tag{12}$$

Embedding into high-dimensional space

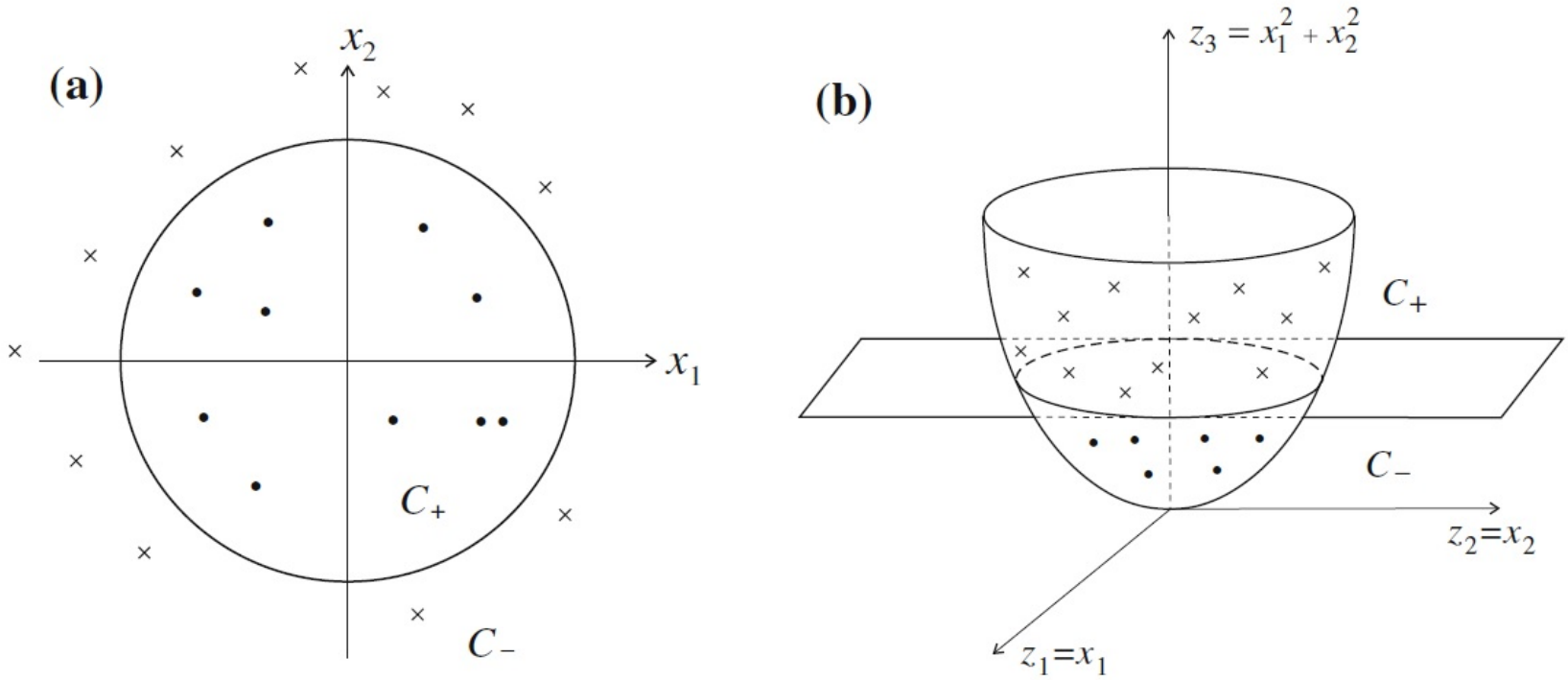

Fig. 11.7 **a** Non-separable in R_2; **b** separable in R^3

Kernel method

Consider the inner product of $\boldsymbol{z} = \varphi(\boldsymbol{x})$ and $\boldsymbol{z}' = \varphi(\boldsymbol{x}')$ after embedding,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{z}(\boldsymbol{x}) \cdot \boldsymbol{z}(\boldsymbol{x}') = \sum z_i(\boldsymbol{x}) z_i(\boldsymbol{x}') \tag{13}$$

We call K as a kernel and then the optimal output function can be written as

$$f(\boldsymbol{x}, \boldsymbol{w}) = \sum \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b, \tag{14}$$

which means we don't need to compute the inner product of φ.

Kernel method

We may start from a kernel function without specifying embedding funcitons, provides $K(\boldsymbol{x}, \boldsymbol{x}')$ is positive-definite satisfying

$$\sum c_i c_j K(\boldsymbol{x_i}, \boldsymbol{x_j}) > 0, \forall \boldsymbol{c} = (c_1, \cdots, c_l), \tag{15}$$

called the Mercer condition.

- **Gaussian kernel**

$$K(\boldsymbol{x}, \boldsymbol{x}') = exp\Big\{ - \frac{|\boldsymbol{x} - \boldsymbol{x}'|^2}{\sigma^2} \Big\} \tag{16}$$

- **Polynomial kernel**

$$K(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} \cdot \boldsymbol{x}' + 1)^p \tag{17}$$

Riemannian metric induced by kernel

The original space R^n of patterns is embedded in R^m, possibly in R^∞, as a curved n-dimensional submanifold.

$$ds^2 = |\varphi(\boldsymbol{x} + d\boldsymbol{x}) - \varphi(\boldsymbol{x})|^2 = \sum \frac{\partial}{\partial x_i}\varphi(\boldsymbol{x}) \cdot \frac{\partial}{\partial x_j}\varphi(\boldsymbol{x}) dx_i dx_j \tag{18}$$

Therefore, the induced Riemannian metric is given by

$$g_{ij}(\boldsymbol{x}) = \left(\frac{\partial}{\partial x_i}\varphi(\boldsymbol{x})\right) \cdot \left(\frac{\partial}{\partial x_j}\varphi(\boldsymbol{x})\right) \tag{19}$$

Riemannian metric induced by kernel

Rewrite it in terms of the kernel as

$$g_{ij}(\boldsymbol{x}) = \frac{\partial^2}{\partial x_i \partial x_j^{'}} K(\boldsymbol{x}, \boldsymbol{x}^{'})|_{\boldsymbol{x}'=\boldsymbol{x}} \tag{20}$$

The volume element at point $\boldsymbol{x}$ is given by

$$dV(\boldsymbol{x}) = \sqrt{|g_{ij}(\boldsymbol{x})|} dx_1 \cdots dx_n, \tag{21}$$

which shows how the volume is enlarged or contracted at around $\boldsymbol{x}$.

Geometric insight

To increase the margin or separability of classes, we need to enlarge the spatial resolution around the boundary $f(\boldsymbol{x}) = 0$, which is expressed by

$$ds^2 = \sum g_{ij} dx_i dx_j.$$

In practice, we don't know the boundary, so by using the knowledge that support vectors are (mostly) located around the boundary, we solve the problem by increasing the metric in the neighborhood of the support vectors.

Conformal transformation

A conformal transformation

$$\tilde{g}_{ij}(\boldsymbol{x}) = \sigma(\boldsymbol{x}) g_{ij}(\boldsymbol{x}), \tag{22}$$

gives a solution to this problem, for the metric is enlarged by a factor $\sigma(\boldsymbol{x})$, which should have large values at the SV positions. In practice, it's difficult to find a mapping $\tilde{\varphi}$ which brings the above conformal transformation. Therefore, we consider a quasi-conformal transformation obtained from modification of a kernel.

Conformal transformation of kernel

Definition (Conformal transformation of kernel)

Let $\sigma(\boldsymbol{x})$ be a positive scalar function.

$$\tilde{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma(\boldsymbol{x})\sigma(\boldsymbol{x}')K(\boldsymbol{x}, \boldsymbol{x}') \tag{23}$$

is called a conformal transformation of a kernel by factor $\sigma(\boldsymbol{x})$.

In Amari and Wu(1999),

$$\sigma(\boldsymbol{x}) = \sum_i e^{-\kappa_i |\boldsymbol{x} - \boldsymbol{x}_i^*|} \tag{24}$$

was chosen, where $\boldsymbol{x}_i^*$ are the SV and κ_i are adequate constants.

Conformal transformation of kernel

$$\begin{aligned}\tilde{g}_{ij}(\boldsymbol{x}) = \sigma^2(\boldsymbol{x})g_{ij}(\boldsymbol{x}) + \sigma_i(\boldsymbol{x})\sigma_j(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{x}) \\ + \sigma(\boldsymbol{x})\{\sigma_i(\boldsymbol{x})K_j(\boldsymbol{x},\boldsymbol{x}) + \sigma_j(\boldsymbol{x})K_i(\boldsymbol{x},\boldsymbol{x})\},\end{aligned} \tag{25}$$

where

$$\sigma_i = \frac{\partial}{\partial x_i}\sigma(\boldsymbol{x}), \quad K_i(\boldsymbol{x},\boldsymbol{x}) = \frac{\partial}{\partial x_i}K(\boldsymbol{x},\boldsymbol{x}')|_{\boldsymbol{x}'=\boldsymbol{x}}. \tag{26}$$

When $K_i(\boldsymbol{x},\boldsymbol{x}) = 0$, which is satisfied by the Gaussian kernel, we have a simplified expression

$$\tilde{g}_{ij}(\boldsymbol{x}) = \sigma^2(\boldsymbol{x})g_{ij}(\boldsymbol{x}) + \sigma_i(\boldsymbol{x})\sigma_j(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{x}). \tag{27}$$

Boosting

Boosting refers to a general and provably effective method of producing a very accurate prediction rule by combining rough and moderately inaccurate rules of thumb.

Boosting

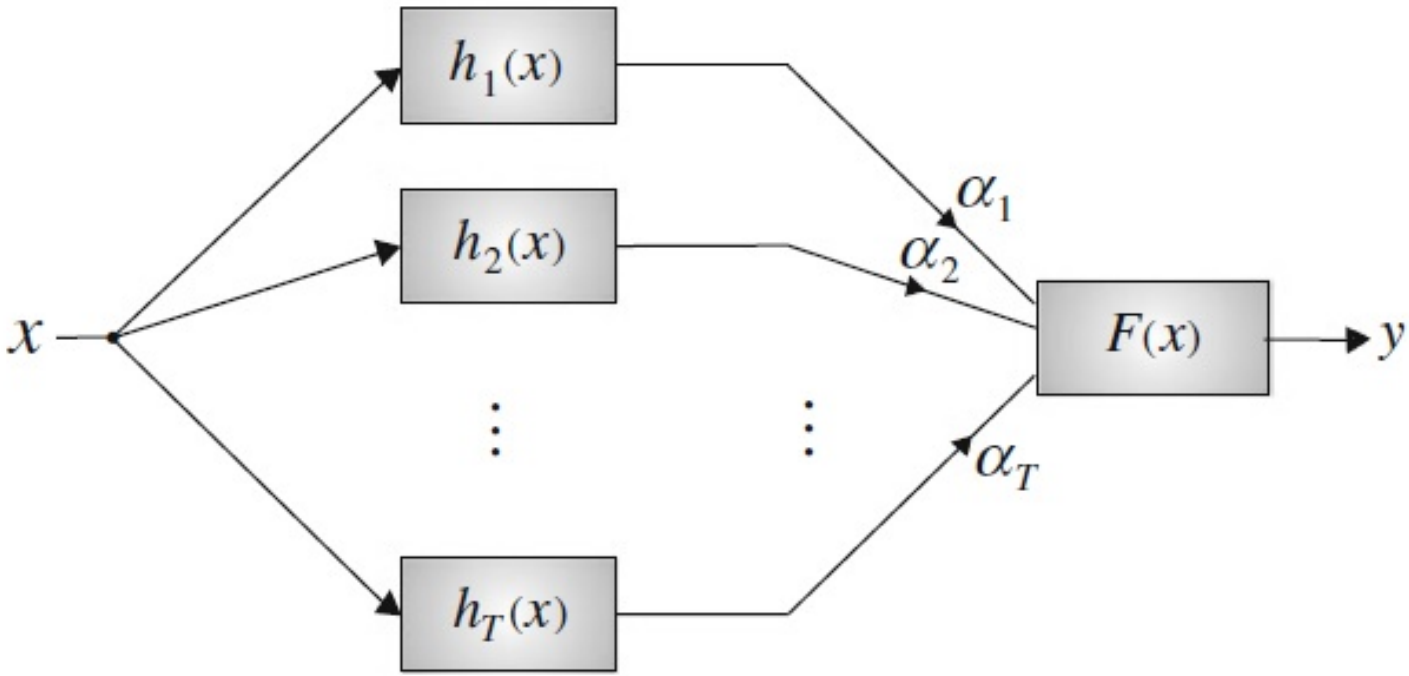

Figure: Integration of weak machines

AdaBoost

The **AdaBoost** algorithm is a famous algorithm in solving the boosting problem. The algorithm takes as input a training set $(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_m, y_m)$ where each $\boldsymbol{x}_i$ belongs to some domain $\mathcal{X}$ and each **label** y_i is in some label set $\mathcal{Y}$, and mostly, we use $\mathcal{Y} = \{-1, +1\}$.

AdaBoost generates weak machines in a series of rounds $t = 1, \cdots, T$. One main idea of the algorithm is to maintain a distribution or set of weights over the training set.

AdaBoost

Algorithm (AdaBoost)

Given $(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_m, y_m)$, *where* $\boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathcal{Y} = \{-1, +1\}$ *Initialize* $D_1(i) = 1/m$. *For* $t = 1, \cdots, T$:

- *Train weak learner* h_t *using distribution* D_y.
- *Choose* $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$, *where* $\epsilon_t = P_{i \sim D_t}[h_t(\boldsymbol{x}_i) \neq y_i]$.
- *Update:*

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(\boldsymbol{x}_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(\boldsymbol{x}_i) \neq y_i \end{cases}$$

$$= \frac{D_t(i) exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{Z_t}$$

where Z_t *is a normalization factor.*

Output the final hypothesis: $H(\boldsymbol{x}) = sign(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}))$.

Measure manifold and probability manifold

$$\mathcal{M} = \left\{ m(y|\boldsymbol{x}) | \sum_{y \in \mathcal{Y}} m(y|\boldsymbol{x}) < \infty (a.e.\boldsymbol{x}) \right\} \tag{28}$$

$$\mathcal{P} = \left\{ m(y|\boldsymbol{x}) | \sum_{y \in \mathcal{Y}} m(y|\boldsymbol{x}) = 1 (a.e.\boldsymbol{x}) \right\} \tag{29}$$

Empirical conditional probability

For given examples $\{(\boldsymbol{x}_i, y_i); i = 1, \cdots, n\}$, let

$$\tilde{p}(y|\boldsymbol{x}) = \begin{cases} \delta(y_i, y), & \boldsymbol{x} = \boldsymbol{x}_i, \\ \dfrac{1}{|\mathcal{Y}|}, & \text{otherwise} \end{cases} \tag{30}$$

be the empirical conditional probability density. Suppose a unique label y_i is given for each input $\boldsymbol{x}_i$.

KL divergence

For two points $p, q \in \mathcal{M}$, the KL divergence extended over $\mathcal{M}$ is defined by

$$D(p,q) = \int_{\mathcal{X}} \mu(\boldsymbol{x}) \sum_{y \in \mathcal{Y}} \left(p(y|\boldsymbol{x}) log \frac{p(y|\boldsymbol{x})}{q(y|\boldsymbol{x})} - p(y|\boldsymbol{x}) + q(y|\boldsymbol{x}) \right) d\boldsymbol{x} \tag{31}$$

where $\mu(\boldsymbol{x})$ is the marginal distribution of $\boldsymbol{x}$, and in the most cases of the following discussion, we fix $\mu(\boldsymbol{x})$ with the empirical distribution $\mu(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} \delta(\boldsymbol{x}_i, \boldsymbol{x})$.

Probability families

Given a set functions $\boldsymbol{f} = \{f_t(\boldsymbol{x}, y); t = 1, \cdots, T\}$ on $\mathcal{X} \times \mathcal{Y}$ and let $\tilde{f}(\boldsymbol{x}) = E_{\tilde{p}}(f|\boldsymbol{x}) = \sum_{y \in \mathcal{Y}} \tilde{p}(y|\boldsymbol{x}) f(\boldsymbol{x}, y)$.

e-flat subspace

$$\mathcal{Q} = \mathcal{Q}(q_0, \tilde{p}, \boldsymbol{f})$$

$$= \left\{ q \in \mathcal{M} \middle| q(y|\boldsymbol{x}) = q_0(y|\boldsymbol{x}) exp\left(\sum_{t=1}^{T} \alpha_t (f_t(\boldsymbol{x}, y) - \tilde{f}_t(\boldsymbol{x})) \right) \right\}$$

m-flat subspace

$$\mathcal{F} = \mathcal{F}(\tilde{p}, \boldsymbol{f})$$

$$= \left\{ q \in \mathcal{M} \middle| \int_{\mathcal{X}} \mu(\boldsymbol{x}) \sum_{y \in \mathcal{Y}} q(y|\boldsymbol{x}) (f_t(\boldsymbol{x}, y) - \tilde{f}_t(\boldsymbol{x})) d\boldsymbol{x} = 0; \forall t \right\}$$

Flatness

$\tilde{p}(y|\boldsymbol{x}) \in \mathcal{F}$.
We can conclude the flatness of the above two families from the following equations.

$$q_0 exp\left(\beta_1 log\frac{q_1}{q_0} + \beta_2 log\frac{q_2}{q_0}\right) \in \mathcal{Q}$$

$$\beta_1 p_1 + \beta_2 p_2 \in \mathcal{F}$$

$$\forall \beta_1, \beta_2 > 0, p_1, p_2 \in \mathcal{F}, q_1, q_2 \in \mathcal{Q}$$

Geometric structure of KL divergence

We know $D = D^{(-1)}$,therefore we have

$$g_{ij}(\xi) = \int \partial_i l^{(-1)}(x;\xi)\partial_j l^{(1)}(x;\xi)dx$$

where $l^{(-1)}(x;\xi) = p(x,\xi), l^{(1)}(x;\xi) = logp(x;\xi)$.
Hence for any $X, Y \in T_\xi(\mathcal{M})$, $\langle X, Y\rangle_\xi = \int X l^{(-1)} Y l^{(1)} dx$.

Orthogonality

Suppose $q^* \in \mathcal{F} \cap \mathcal{Q}$. For any $p \in \mathcal{F}, q \in \mathcal{Q}$, let $\gamma_1^{(-1)}(x;t) = tq^{(-1)} + (1-t)q^{*(-1)}$ and $\gamma_2^{(1)}(x;t) = tp^{(1)} + (1-t)q^{*(1)}$ be the m-geodesic connecting p, q^* and the e-geodesic connecting q, q^* respectively.

$$\langle \dot{\gamma}_1(0), \dot{\gamma}_2(0) \rangle_{q^*} = \int \frac{d\gamma_1^{(-1)}(x;t)}{dt}\Big|_{t=0} \frac{d\gamma_2^{(1)}(x;s)}{ds}\Big|_{s=0} dx$$

$$= \int (q^{(-1)} - q^{*(-1)})(p^{(1)} - q^{*(1)}) dx$$

$$= \int_{\mathcal{X}} \sum_{y \in \mathcal{Y}} (\sum_{t=1}^{T} \beta_t (f_t(\boldsymbol{x}, y) - \tilde{f}_t(\boldsymbol{x})))(q(y|\boldsymbol{x}) - q^*(y|\boldsymbol{x})) d\mu(x) = 0$$

Thus, $\mathcal{F}$ and $\mathcal{Q}$ is orthogonal at q^*.

Pythagorean relation

Lemma

For any $p \in \mathcal{F}(\tilde{p}, \boldsymbol{f})$ and $q \in \mathcal{Q}(q_0, \tilde{p}, \boldsymbol{f})$, the Pythagorean relation

$$D(p, q) = D(p, q^*) + D(q^*, q) \tag{32}$$

holds.

We have immediately that $\{q^*\} = \mathcal{F} \cap \mathcal{Q}$.

Theorem (Lebanon and Lafferty (2001))

$$q^* = \underset{p \in \mathcal{F}(\tilde{p}, \boldsymbol{f})}{\operatorname{argmin}} D(p, q_0) = \underset{q \in \mathcal{Q}(q_0, \tilde{p}, \boldsymbol{f})}{\operatorname{argmin}} D(\tilde{p}, q) \tag{33}$$

Pythagorean relation

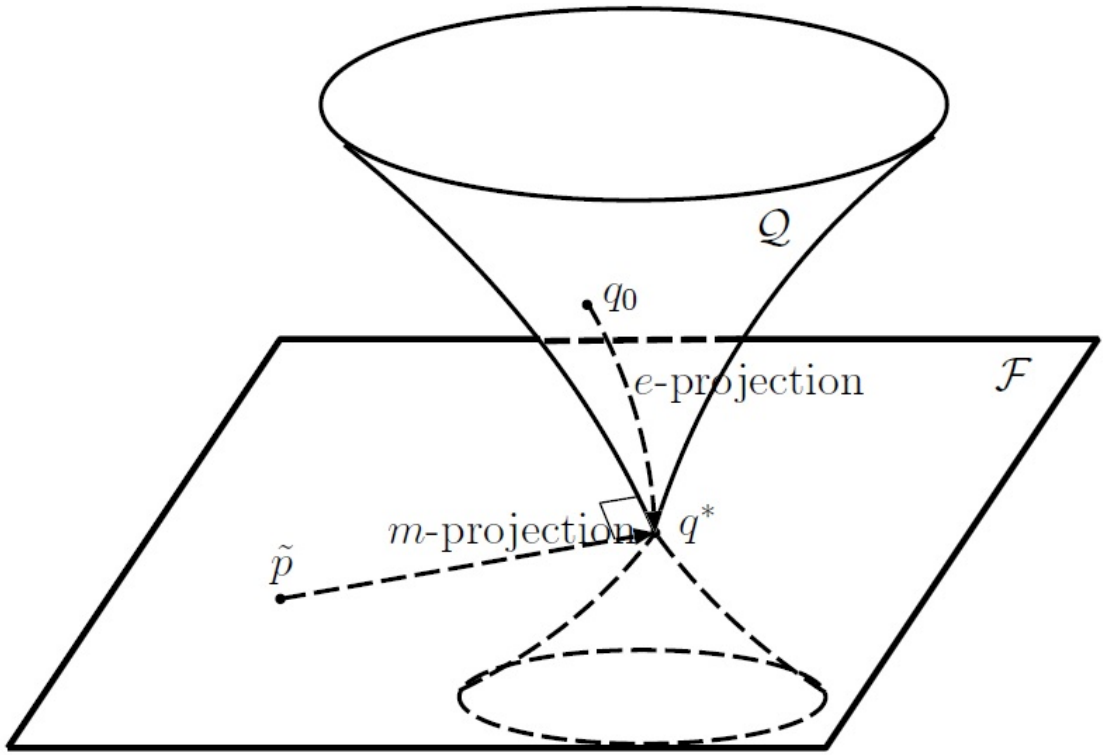

Figure: A geometrical interpretation of the dual optimization problems in the AdaBoost algorithm.

Geometric view of AdaBoost

$$\begin{aligned}\mathcal{Q}_t &= \mathcal{Q}(q_{t-1}, \tilde{p}_t, f_t) \\ &= \left\{ q \in \mathcal{M} \middle| q(y|\boldsymbol{x}) = q_{t-1}(y|\boldsymbol{x}) exp(\alpha_t(f_t(\boldsymbol{x}, y) - \tilde{f}_t(\boldsymbol{x}))) \right\}\end{aligned}$$

$$\begin{aligned}\mathcal{F}_t &= \mathcal{F}(\tilde{p}, f_t) \\ &= \left\{ p \in \mathcal{M} \middle| \int_{\mathcal{X}} \mu(\boldsymbol{x}) \sum_{y \in \mathcal{Y}} p(y|\boldsymbol{x})(f_t(\boldsymbol{x}, y) - \tilde{f}_t(\boldsymbol{x}))d\boldsymbol{x} = 0 \right\}\end{aligned}$$

It can be verified that $\mathcal{Q}_t$ and $\mathcal{F}_t$ intersect at one point q_t.

Geometric view of AdaBoost

Algorithm (AdaBoost)

1. *Initialize* $q_0(y|\boldsymbol{x}) = 1$.
2. *For* $t = 1, \cdots, T$
   - *Select a machine* h_t *s.t.*

$$\sum_{i=1}^{n} \sum_{y \in \mathcal{Y}} q_{t-1}(y|\boldsymbol{x}_i)(f_t(\boldsymbol{x}_i, y) - f_t(\boldsymbol{x}_i, y_i)) \neq 0$$

   where $f_t(\boldsymbol{x}, y) = \begin{cases} 1/2, & y \in h_t(\boldsymbol{x}), \\ -1/2, & \text{otherwise}. \end{cases}$

   - *Construct* $\mathcal{Q}_t$ *and* $\mathcal{F}_t$ *with* $\tilde{p}, q_{t-1}, f_t$.
   - *Find* q_t *and corresponding* α_t *which is the intersection of* $\mathcal{Q}_t$ *and* $\mathcal{F}_t$

Geometric view of AdaBoost

Algorithm (AdaBoost)

3. *Output the final decision as the majority vote of* $\{h_t; t = 1, \cdots, T\}$ *with weights* $\{\alpha_t; t = 1, \cdots, T\}$

$$H(\boldsymbol{x}) = \underset{y \in \mathcal{Y}}{argmax} \sum_{t=1}^{T} \alpha_t f_t(\boldsymbol{x}, y)$$

For the binary case, $\alpha_t = \frac{1}{2} log \frac{1-\epsilon_t}{\epsilon_t}$, where ϵ_t is the weighted error defined by $\epsilon_t = \sum_{i=1}^{n} I(y_i \neq h_t(\boldsymbol{x}_i)) D_t(i)$, $D_t(i) = \frac{q_{t-1}(y \neq y_i | \boldsymbol{x}_i)}{Z_t}$, and Z_t is a normalization constant.

Geometric view of AdaBoost

In fact since the relation $D(\tilde{p}, q_{t-1}) = D(\tilde{p}, q_t) + D(q_t, q_{t-1})$ holds, as t increases q_t approaches to $\tilde{p}$ as long as $D(q_t, q_{t-1}) > 0$.

$$\begin{aligned} H(\boldsymbol{x}) &= \underset{y \in \mathcal{Y}}{argmax} \sum_{t=1}^{T} \alpha_t f_t(\boldsymbol{x}, y) \\ &= \underset{y \in \mathcal{Y}}{argmax} \log q_T(y|\boldsymbol{x}) \\ &= \underset{y \in \mathcal{Y}}{argmax}\, q_T(y|\boldsymbol{x}) \end{aligned}$$

Geometric view of AdaBoost

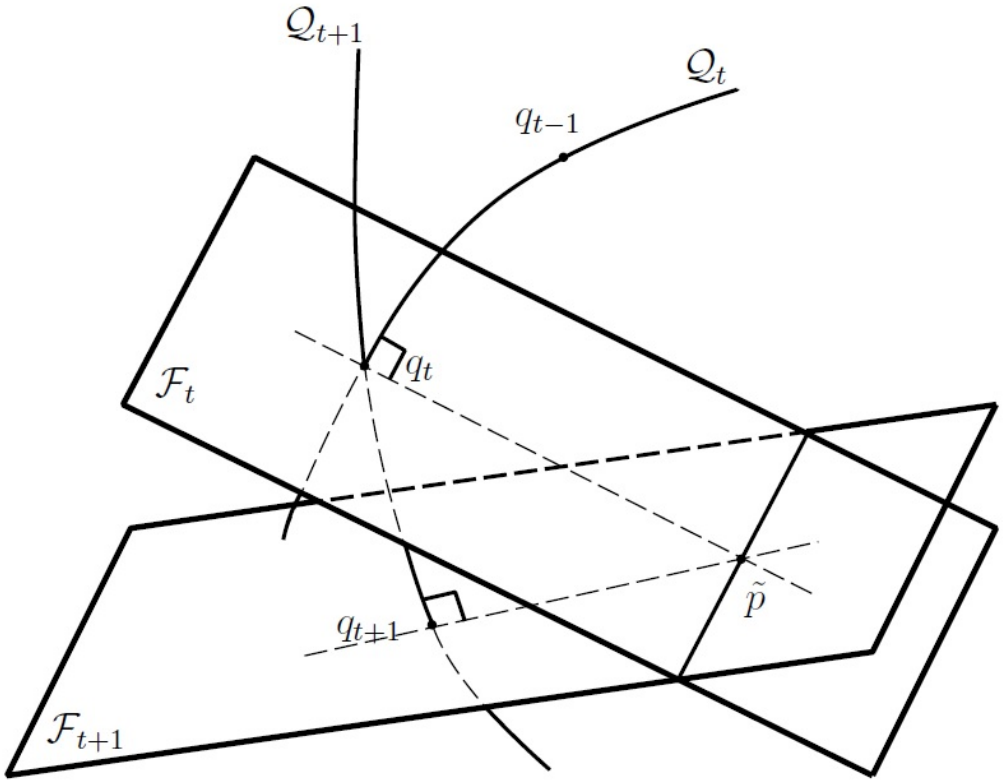

Figure: A geometrical interpretation of the sequential update of AdaBoost.

U-boost

Using Bregman divergence of convex function U (in the KL divergence, $U(z) = exp(z)$) $D_U(p, q)$ instead of KL divergence, we can obtain a series of new boost algorithms.