Anàlisi d'Imatges i Reconeixement de Formes

Image Analysis and Pattern Recognition:

3. Linear Machines and Neurons

Francesc J. Ferri

Dept. d'Informàtica. Universitat de València

Gener 2008

Problem definition

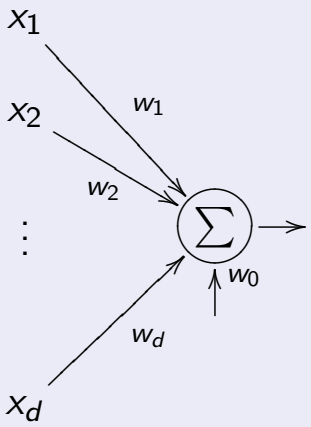
Instead of assuming a parametric model for the underlying statistical structure of data or looking for a convenient distance or dissimilarity measure among data, a **parametric model** for the corresponding **discriminant functions** can be imposed to solve the pattern recognition problem.

$$g(x; W)$$

The parameters W are usually called **weights** in this context and a convenient criterion function, $J(W)$ must be defined in order to look for W.

Linear Models for Classification

One of the simplest and most used cases is **linear** discriminant functions



$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

Linear Models

One can also compute a **fixed** mapping of the input instead

$$g(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + w_0$$

And write it in a compact form as $\boxed{g(\mathbf{x}) = \mathbf{a}^T\mathbf{y}(\mathbf{x})}$

where $\mathbf{a}^T = (w_0, \mathbf{w}^T)$ is the weight vector and $\mathbf{y}(\mathbf{x})^T = (1, \phi(\mathbf{x})^T)$

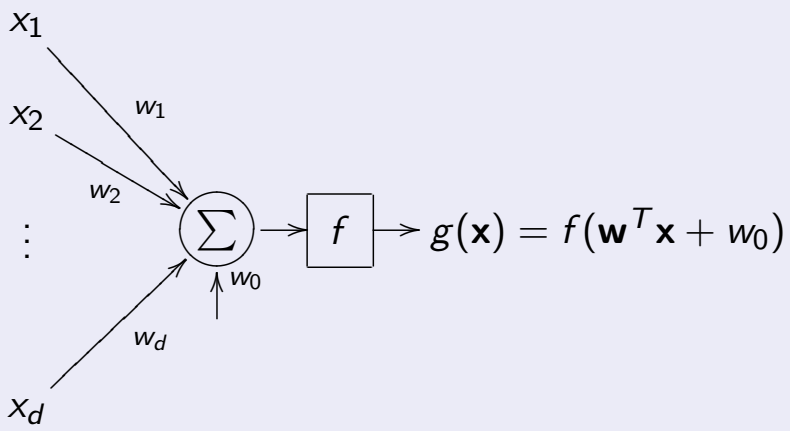
In general, ϕ is a vectorial mapping whose dimensionality may be different from d, as in

$$g(\mathbf{x}) = w_0 + \mathbf{w}^T\mathbf{x} + \mathbf{x}^T W\mathbf{x}$$

$\mathbf{y}$ are usually referred to as **extended vectors**

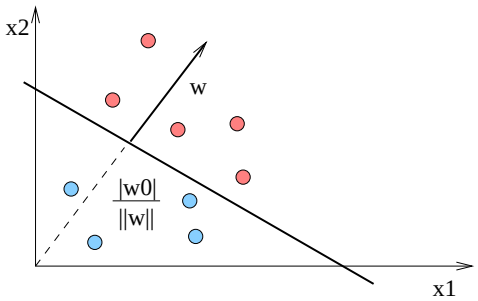
Linear models

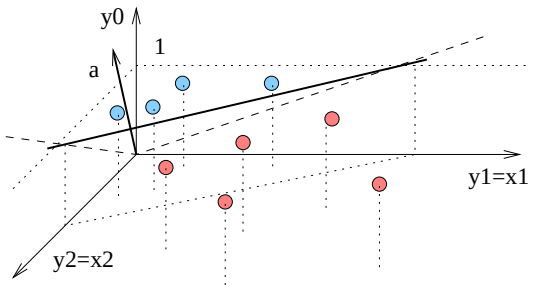
The decision boundaries correspond to $g(x) = const.$ so they do not get modified if an **activation function** is applied. So in general,



Linear Models. Geometry

$g(\mathbf{x}) = 0$ defines an hyperplane whose director vector is $\mathbf{w}$ and whose distance to the origin is given by $\frac{|w_0|}{||\mathbf{w}||}$.



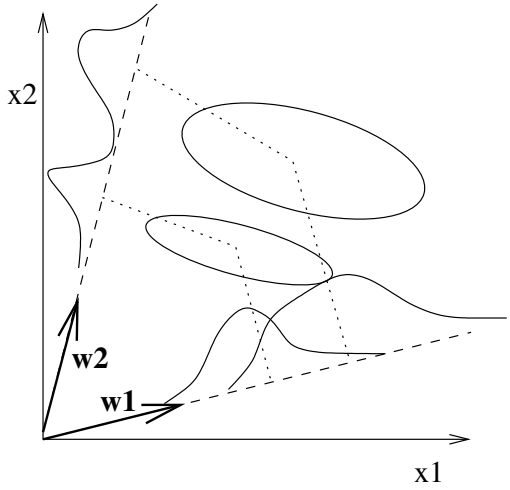


In general, the signed (Euclidean) distance from any $\mathbf{x}$ to the hyperplane is given by $\frac{g(\mathbf{x})}{||\mathbf{w}||}$ and its magnitude $\frac{|g(\mathbf{x})|}{||\mathbf{a}||} \leq \frac{|g(\mathbf{x})|}{||\mathbf{w}||}$.

Fisher's linear discriminant

A linear discriminant for a 2-class problem can be decomposed as a mapping from $\mathbb{R}^d$ to $\mathbb{R}$ plus a threshold.

It is possible to try to define a mapping $\mathbf{w}$ which maximizes separability between classes.



Fisher's linear discriminant

Class separability can be related to (projected) class mean distance, $\Delta\tilde{\mu} = \tilde{\mu}_1 - \tilde{\mu}_2$ or scatter, $\Delta\tilde{\mu}^T\Delta\tilde{\mu}$, where $\tilde{\mu}_i = \mathbf{w}\mu_i$.

But mean distances need to be combined with (low) class scatter (or variance), $\tilde{s}_1 + \tilde{s}_2$, where $\tilde{s}_i = \mathbf{w}^T s_i \mathbf{w}$.

Fisher's criterion (2-class)

$$J(\mathbf{w}) = \frac{\Delta\tilde{\mu}^2}{\tilde{s}_1 + \tilde{s}_2} = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

where $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$ and
$S_W = \sum_{i=1,2} S_i = \sum_{i=1,2} \sum_{\mathbf{x}\in D_i} (\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T$.

Fisher's linear discriminant (2-class)

The projection that maximizes J_F is given can be obtained by $\nabla_w J_F = 0$ which implies that

$$S_W \mathbf{w} = S_B \mathbf{w} \frac{\mathbf{w}^T S_W \mathbf{w}}{\mathbf{w}^T S_B \mathbf{w}} \propto S_B \mathbf{w} \propto \Delta\mu$$

For non singular S_W one can safely project data using

$$\mathbf{w} = S_W^{-1} \Delta\mu$$

Once the data is projected onto 1D, any classifier can be designed. For example, parametric estimation of (normal) Bayes classifier becomes feasible.

Fisher's linear discriminant (multiple classes)

In general we can define the total, within and between scatter matrices:

$$S_T = \sum_{\mathbf{x}} (\mathbf{x} - \mu)(\mathbf{x} - \mu)^T$$

$$S_W = \sum_{i=1}^{c} S_i$$

$$S_B = \sum_{i=1}^{c} n_i (\mu_i - \mu)(\mu_i - \mu)^T$$

such that

$$S_T = S_W + S_B$$

Fisher's linear discriminant (multiple classes)

Instead of a vector $\mathbf{w}$ we consider now a matrix W (a linear mapping from $\mathbb{R}^d$ to $\mathbb{R}^{d'}$) and $W^T S$ is a projected d'-dimensional scatter matrix.

The Fisher criterion can be defined either as

$$J_F(W) = \frac{|W^T S_B W|}{|W^T S_W W|}$$

or

$$J_F(W) = \frac{Tr\{W^T S_B W\}}{Tr\{W^T S_W W\}}$$

Fisher's linear discriminant (multiple classes)

It can be shown that the columns of the optimal W consist of the generalized eigenvectors that correspond to largest eigenvalues in

$$S_B \mathbf{w}_i = \lambda_i S_W \mathbf{w}_i$$

Also, for nonsingular S_W, the eigenvectors of $S_W^{-1} S_B$ can be computed instead.

Remark:

as S_B is at most of rank $c-1$ (sum of c one-rank matrices constrained by the fact that μ is a linear combination of μ_i), the maximum number of columns in optimal W is $c-1$.

The Perceptron

Let $\Omega = \{\omega_1, \omega_2\}$, $\mathbf{a} = (w_0, w_1, \ldots, w_d)^T$ and $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ where $\mathbf{x}_i \in \mathbb{R}^d$. Define

$$z_i = \begin{cases} 1 & \text{if } \mathbf{x}_i \in \omega_1 \\ -1 & \text{if } \mathbf{x}_i \in \omega_2 \end{cases}$$

In this way, all samples in D will be correctly classified if and only if (in terms of extended vectors and weights)

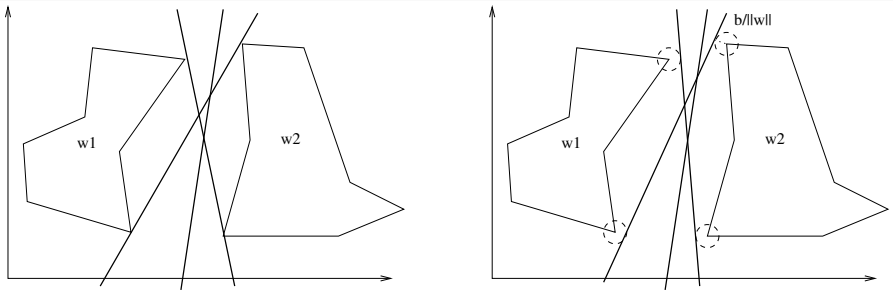
$$z_i \mathbf{a}^T \mathbf{y}_i > 0 \quad \forall i$$

Linear separability

If such a vector $\mathbf{a}$ exists, D is said to be linearly separable.

The linear discrimination problem

The problem consists of finding a vector solution, $\mathbf{a}$, for a system of linear constraints. The additional constraint $||\mathbf{a}|| = 1$ can be used.



An alternative (stricter) formulation of the problem (using margins) is

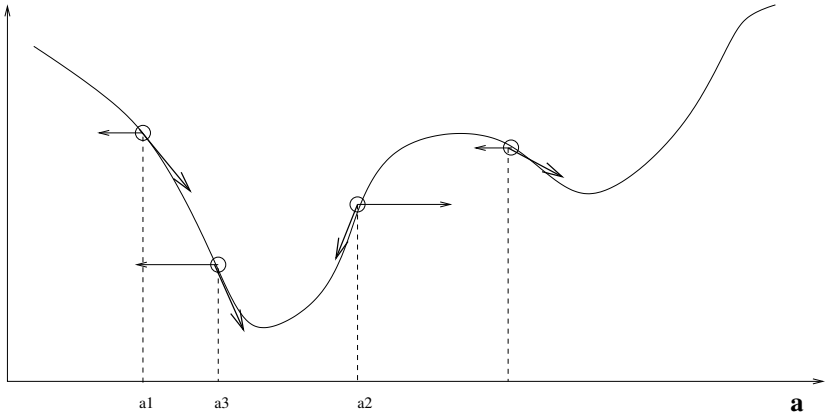
$$\text{Minimize: } ||\mathbf{a}|| \text{ subject to: } z_i \mathbf{a}^T y_i > b$$

$b/||\mathbf{w}||$ is the minimum distance from any point to the separating hyperplane.

Gradient descent

Let $J(\mathbf{a})$ be a convenient criterion function satisfying very mild smoothness conditions.

$J(\mathbf{a})$ is a scalar function that can be seen as a surface over $\mathbb{R}^{d+1}$. The gradient, $\nabla J(\mathbf{a})$ is a vector in $\mathbb{R}^{d+1}$ whose magnitude accounts for the local slope of the surface and points to the direction of maximal (positive) change.



Gradient descent

Gradient descent algorithm

$$\mathbf{a}_0 = \text{arbitrary}$$

$$\mathbf{a}_{k+1} = \mathbf{a}_k - \eta(k)\nabla J_p(\mathbf{a})$$

The learning rate $\eta(k)$ is a function of k and controls the amount of change to apply at each iteration.

The learning rate is usually set to an appropriate constant or is set as $\eta(k) = \frac{\eta(0)}{k}$.

Under certain circumstances the convergence of the algorithm to a local (or even global) minimum can be established.

Extensions to basic gradient descent

Consider Second order expansion of J around $\mathbf{a}_0$

$$J(\mathbf{a}) = J(\mathbf{a}_0) + \nabla J^T(\mathbf{a} - av_0) + \frac{1}{2}(\mathbf{a} - av_0)^T H(\mathbf{a} - av_0)$$

where H is the Hessian matrix, $\{\frac{\partial^2 J}{\partial a_i \partial a_j}\}$.

Assuming smoothness and $\mathbf{a}(k+1) - \mathbf{a}(k) = -\eta(k)\nabla J$

$$J(\mathbf{a}(k+1)) \approx J(\mathbf{a}(k)) - \eta(k)\nabla J^T \nabla J + \frac{1}{2}\eta(k)^2 \nabla J^T H \nabla J$$

This is minimized when $[\eta\nabla^2 - \frac{1}{2}\eta^2 \nabla H \nabla]$ is maximized.
Deriving with regard to η will give the optimal rate as $\eta = \frac{\nabla^2}{\nabla H \nabla}$.

Second order gradient descent

By setting an optimal rate an improved descent algorithm is obtained.

2nd order gradient descent algorithm

$$\mathbf{a}_{k+1} = \mathbf{a}_k - \frac{||\nabla J||^2}{\nabla J^T H \nabla J} \nabla J_p(\mathbf{a})$$

Alternatively, the first and 2nd order terms in the expansion of J can be minimized by deriving with respect to $\mathbf{a}$ and equating to 0.

Newton method

$$\mathbf{a}_{k+1} = \mathbf{a}_k - H^{-1} \nabla J_p(\mathbf{a})$$

The Perceptron Criterion

A good criterion to measure how good a linear classifier is could consist of the total number of misclassified samples.

A (piecewise) differentiable version of the above is the sum of the distances to the separating hyperplane for misclassified samples

$$J_p(\mathbf{a}) = \sum_{i:z_i\mathbf{a}^T\mathbf{y}_i \leq 0} (-\mathbf{a}^T\mathbf{y}_i)$$

The corresponding gradient is

$$\nabla J_p(\mathbf{a}) = \sum_{i:z_i\mathbf{a}^T\mathbf{y}_i \leq 0} (-\mathbf{y}_i)$$

Perceptron algorithm

The preceptron criterion can be minimized using a gradient descent approach starting from an arbitrary classifier given by $\mathbf{a}_0$

Batch perceptron algorithm

$$\mathbf{a}_{k+1} = \mathbf{a}_k - \eta(k) \sum_{i:z_i\mathbf{a}^T\mathbf{y}_i \leq 0} (-\mathbf{y}_i)$$

As the criterion J_p is a piecewise linear function, the convergence to a solution for linear separable sets can be taken for granted.

Batch versus per sample correction

A closely related criterion involving only one sample can be defined as $J_p(\mathbf{a}, \mathbf{y}_i) = -\mathbf{a}^T\mathbf{y}_i$ if $z_i\mathbf{a}^T\mathbf{y}_i \leq 0$ and zero otherwise. Then

Single-sample perceptron algorithm

$$\mathbf{a}_{k+1} = \mathbf{a}_k + \eta(k)\mathbf{y}_i$$

Other, more or less related criteria can be defined

$$J_q(\mathbf{a}) = \frac{1}{2}\sum_{i:z_i\mathbf{a}^T\mathbf{y}_i\leq 0}(\mathbf{a}^T\mathbf{y}_i)^2 \qquad \nabla J_q(\mathbf{a}) = \sum_{i:z_i\mathbf{a}^T\mathbf{y}_i\leq 0}(\mathbf{a}^T\mathbf{y}_i)\mathbf{y}_i$$

$$J_r(\mathbf{a}) = \frac{1}{2}\sum_{i:z_i\mathbf{a}^T\mathbf{y}_i\leq 0}\frac{(\mathbf{a}^T\mathbf{y}_i - b)^2}{||\mathbf{y}_i||^2} \qquad \nabla J_r(\mathbf{a}) = \sum_{i:z_i\mathbf{a}^T\mathbf{y}_i\leq 0}\frac{(\mathbf{a}^T\mathbf{y}_i - b)}{||\mathbf{y}_i||}\mathbf{y}_i$$

Learning using Least Minimum Squares

Instead of looking for a solution to a system or linear constraints $z_i\mathbf{a}^T\mathbf{y}_i > 0$, it is possible to specify a set of **desired outputs**, b_i, and try to solve a set of linar equations, $\mathbf{a}^T\mathbf{y}_i = b_i$.

In matrix form,

$$Y\mathbf{a} = \mathbf{b}$$

where $\mathbf{b} = (b_1, \ldots, b_N)^T$ and $Y = (\mathbf{y}_1^T, \ldots, \mathbf{y}_N^T)^T$.

It is possible to solve this by least squares by setting as a criterion:

$$J_s(\mathbf{a}) = ||Y\mathbf{a} - \mathbf{b}||^2 = \sum_{i=1}^{N}(\mathbf{a}^T\mathbf{y}_i - b_i)^2$$

The pseudoinverse method

The least square solution is given by equating the gradient of the criterion to zero

$$\nabla J_s = \sum_{i=1}^{N} 2(\mathbf{a}^T \mathbf{y}_i - b_i)\mathbf{y}_i = 2Y^T(Y\mathbf{a} - \mathbf{b}) = 0$$

which implies

$$T^T Y\mathbf{a} = Y^T\mathbf{b}$$

when $Y^T Y$ is non singular we have obtain the **pseudoinverse** method

$$\mathbf{a} = (Y^T Y)^{-1} Y^T \mathbf{b} = Y^+ \mathbf{b}$$

The Widrow-Hoff Adaline

An alternative way of minimizing the same square error criterion is by gradient descent.

Batch Widrow-Hoff algorithm

$$\mathbf{a}_{k+1} = \mathbf{a}_k - \eta(k)\nabla J_s(\mathbf{a}) = \mathbf{a}_k - \eta(k)Y^T\Delta$$

where $\Delta = Y\mathbf{a} - \mathbf{b}$ is a vector containing the current prediction errors.

Single-sample Widrow-Hoff algorithm

$$\mathbf{a}_{k+1} = \mathbf{a}_k - \eta(k)(\mathbf{a}^T\mathbf{y}_i - b_i)\mathbf{y}_i = \mathbf{a}_k - \eta(k)\delta_i\mathbf{y}_i$$

where $\delta_i = \mathbf{a}^T\mathbf{y}_i - b_i$ is the i-th current prediction error.

This is also known as Adaptive linear element (Adaline), Least-mean-squared (LMS) or **delta** rule.

Improving LMS for linear separable problems

There is no guarantee of finding an existing linearly separable solution using LMS.

Given Y linearly separable $\Rightarrow \exists \mathbf{a}, \mathbf{b} : Y\mathbf{a} = \mathbf{b}$

The problem consists of setting an appropriate set of desired outputs, $\mathbf{b}$ in advance.

Alternatively, we can try to minimize

$$J_s(\mathbf{a}, \mathbf{b}) = ||Y\mathbf{a} - \mathbf{z}^T\mathbf{b}||^2$$

subject to $\mathbf{b} > 0$. (where $\mathbf{z} = (z_1, \ldots, z_N)$).

Ho-Kashyap method

The corresponding gradients

$$\nabla_{\mathbf{a}} J_s = 2Y^T(Y\mathbf{a} - \mathbf{b})$$

$$\nabla_{\mathbf{b}} J_s = -2(Y\mathbf{a} - \mathbf{b})$$

The problem is that we need to maintain the constraint $\mathbf{b} > 0$ and avoid the (trivial) solution $\mathbf{b} = 0$.

Solution: start with small $\mathbf{b}$ and perform only **positive** corrections.

$$\mathbf{b}(k+1) = \mathbf{b}(k) - \eta(k)(\Delta + |\Delta|)$$

And obtain $\mathbf{a}$ from $\mathbf{b}$ with the pseudoinverse.

$$\mathbf{a}(k+1) = Y^+\mathbf{b}(k+1)$$

Learning using maximal margins

In a linear discriminant with margin b, we want to solve

$$g(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + w_0 > z_i b$$

This means that all points need to be at least at a distance $\frac{b}{||\mathbf{w}||}$ from the separating hyperplane, H.

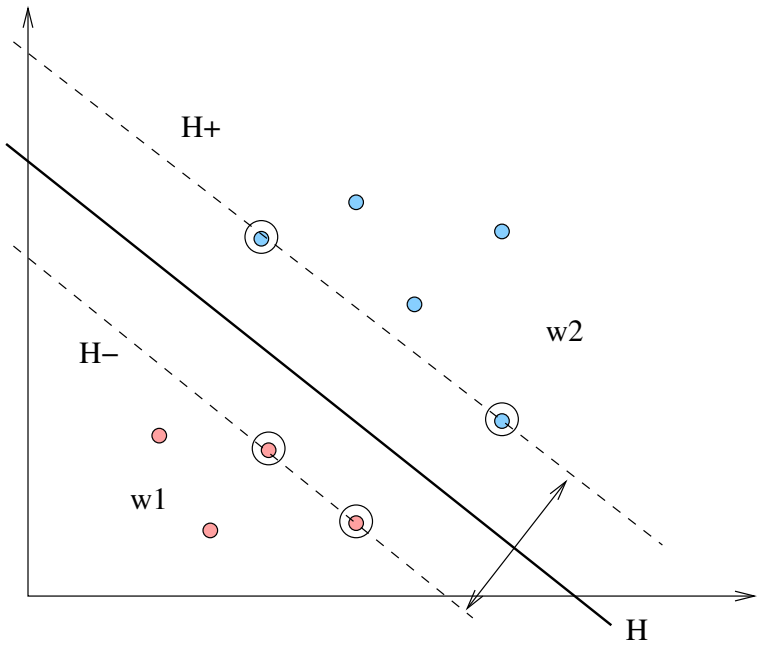
As the particular value of b is not important, we can fix $b = 1$ and try to solve

$$g(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x} + w_0 > z_i$$

and minimize $||\mathbf{w}||$, which implies maximizing all distances to H, and in particular the (minimal) ones from points for which $g(\mathbf{x}_i) = z_i$.

This minimal distance, $\frac{1}{||\mathbf{w}||}$, that has to be maximal is referred to simply as the **margin**.

Learning using maximal margins



The (linear) Support Vector Machine

Formally,

Minimize $||\mathbf{w}||^2$ (= maximal margin)
subject to: $g(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + w_0 \geq z_i \ \forall i$ (= correctly classified)

This is a quadratic optimization problem that can be solved in a number of ways. In particular can convert it into

Minimize: $L_p(\mathbf{w}, w_o, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i(\mathbf{w}^T\mathbf{x}_i + w_o - z_i)$

subject to: $\alpha_i \geq 0$

The α_i's are known as Lagrange multipliers.

The (linear) Support Vector Machine

Definition: suport vector (given $\mathbf{w}$ and w_o)

Any point, $\mathbf{x}$ such that $g(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + w_o = 1$.

Intuitively,

The solution to our problem (that is, $\mathbf{w}$ and w_0) must depend only on support vectors. (We will obtain the very same solution if we remove all other points).

Solving quadratic constrained optimization

The optimal solution must satisfy

$$\nabla_{\mathbf{w}} L_p = \mathbf{w} - \sum_{i=1}^{N} z_i \alpha_i \mathbf{x}_i = 0$$

from which an expression for $\mathbf{w}$ as a linear combination of training points is obtained

$$\mathbf{w} = \sum_{i=1}^{N} z_i \alpha_i \mathbf{x}_i$$

We could now rewrite the problem from the beginning and express the discriminant in terms of α "weights" and w_0:

$$g(\mathbf{x}) = \sum_{i=1}^{N} z_i \alpha_i \mathbf{x}_i^T \mathbf{x} + w_0, \qquad \forall \mathbf{x}$$

Solving quadratic constrained optimization

The minimization of L_p (the primal), can be converted into a completely equivalent but different maximization problem (the dual) which is

Maximize: $L_d = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j z_i z_j \mathbf{x}_i^T \mathbf{x}_j$

subject to: $\sum_{i=1}^{N} z_i \alpha_i = 0$ and $\alpha_i \geq 0 \ \forall i$

The solution for this must satisfy the so-called Karush-Kuhn-Tucker conditions. In this case

Karush-Kuhn-Tucker conditions

$\alpha_i \geq 0$	(obvious)
$z_i g(\mathbf{x}_i) \geq 1$	(correct classification with margin)
$\alpha_i (z_i g(\mathbf{x}_i) - 1) = 0$	($\alpha_i = 0$ iff x_i is not a support vector)

The training of (linear) support vector machines

A quadratic solver is used to obtain α's from the (easier) dual problem.

Then w_o is computed

How? think about it.

The final discriminant function is given in terms of support vectors and α's

$$g(\mathbf{x}) = \sum_{i:\alpha_i>0} z_i \alpha_i \mathbf{x}_i^T \mathbf{x} + w_0$$

Using soft margins

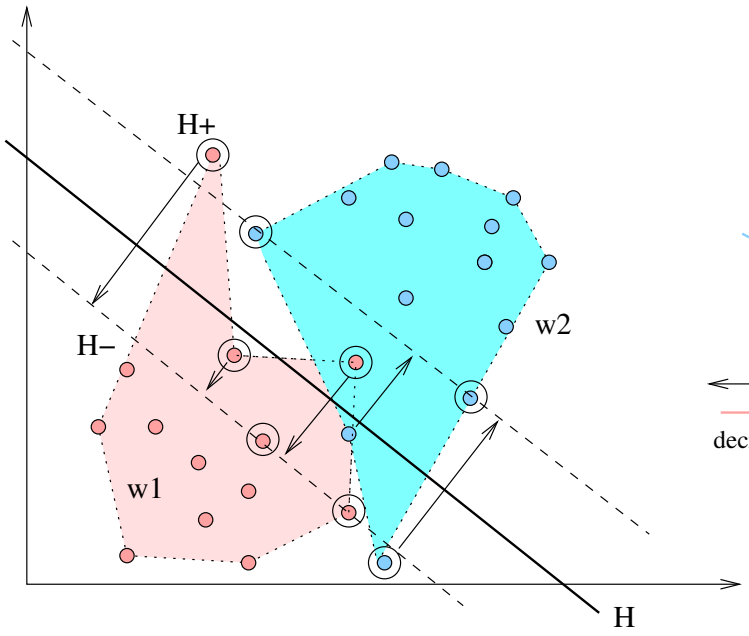
Basic (linear) support vector machines can only solve linearly separable problems.

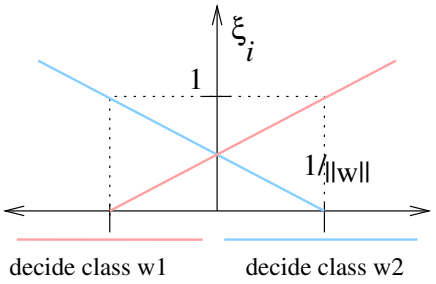
To account for solutions in which some of the points can be (slightly) missclassified, the formulation can be exended by adding a set of **slack** variables.

$$g(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i + w_0 \geq (1 - \xi_i) z_i$$

Clearly if $\xi_i = 0$ the corresponding $\mathbf{x}_i$ has to be outside the margin. Points on either side of the margin will satisfy $0 < \xi \leq 1$ while points outside the margin but on the wrong side will have $\xi > 1$.

Using soft margins





Soft margin support vector machines

The optimization problem can now be stated as

Minimize $||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i$ (= maximal soft margin)
subject to: $g(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + w_0 \geq (1-\xi_i)z_i$ (= correctly (soft) classified)
and $\xi_i \geq 0\ \forall i$

From which the primal now is

Minimize:

$$L_p(\mathbf{w}, w_o, \alpha, \mu) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(z_i(\mathbf{w}^T\mathbf{x}_i + w_o) - 1 + \xi_i) - \sum_{i=1}^{N}\mu_i\xi_i$$

subject to: $\alpha_i \geq 0$ and $\mu_i \geq 0$

where the Lagrange multipliers are now α_i and μ_i.

The training of (linear) support vector machines

A quadratic solver is used to obtain α's from the (easier) dual problem.

Then w_o is computed

How? think about it.

The final discriminant function is given in terms of support vectors and α's

$$g(\mathbf{x}) = \sum_{i:\alpha_i>0} z_i \alpha_i \mathbf{x}_i^T \mathbf{x} + w_0$$

Soft margin support vector machines

Maximize: $L_d = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j z_i z_j \mathbf{x}_i^T \mathbf{x}_j$

subject to: $\sum_{i=1}^{N} z_i \alpha_i = 0$ and $0 \leq \alpha_i \leq C \ \forall i$

Now also from $\frac{\partial L_p}{\partial \xi_i} = C - \alpha_i - \mu_i = 0$ we have

$$\alpha_i = C - \mu_i$$

Karush-Kuhn-Tucker conditions

$\alpha_i \geq 0, \xi_i \geq 0, \mu_i \geq 0$	(obvious)
$z_i g(\mathbf{x}_i) \geq 1 - \xi_i$	(correct classification with soft margin)
$\alpha_i(z_i g(\mathbf{x}_i) - 1 + \xi_i) = 0$	($\alpha_i = 0$ iff x_i is not a support vector)
$\mu_i \xi_i = 0$	(either μ_i or ξ_i equals 0.

The training of (linear) support vector machines with soft margin

A quadratic solver is used to obtain α's and μ's from the dual problem and w_0 is obtained from them.

New definition: suport vector (given $\mathbf{w}$, w_o and ξ_i)

Any point, $(\mathbf{x}_i, z_i)$ such that $z_i g(\mathbf{x}_i) = z_i(\mathbf{w}^T \mathbf{x}_i + w_o) = 1 - \xi_i$.

The final discriminant function is given in terms of support vectors and α's

$$g(\mathbf{x}) = \sum_{i:\alpha_i>0} z_i \alpha_i \mathbf{x}_i^T \mathbf{x} + w_0(\alpha, \mu)$$

Support vector machines with soft margin

There are two kinds of support vectors ($\mathbf{x}_i$ such that $\alpha_i > 0$):

- $0 < \alpha_i < C$ which implies $\mu_i > 0$ and consequently $\xi_i = 0$ which means the support vector is exactly at distance $\frac{1}{||\mathbf{w}||}$ from the separating hyperplane.
- $\alpha_i = C$ which implies $\mu_i = 0$ and as there is no restriction on the value of ξ_i:
  - $\xi_i = 0$: at distance $\frac{1}{||\mathbf{w}||}$ as before.
  - $0 < \xi_i \leq \frac{1}{2}$ or $\frac{1}{2} \leq\leq 1$: inside the margin, either on the right or wrong side.
  - $\xi_i \geq 1$: the support vector is further than $\frac{1}{||\mathbf{w}||}$ on the wrong side.

Support vector machines: practical considerations

The trade-off parameter, C, controls the relative importance given to misclassifications. For $C = \infty$ we obtain the basic support vector solution.

It is very difficult to set an appropriate value for C a priori. In practice one usually selects this by a cross validation evaluation test.

There is an alternative formulation of soft-margin SVMs which are called ν-SVMs in which the (meaningless) parameter C is substituted by ν which can be interpreted as an upper bound on the fraction of **margin errors** and as a lower bound on the fraction of support vectors.