Linear classifiers, a perceptron family

Václav Hlaváč

Czech Technical University in Prague

Czech Institute of Informatics, Robotics and Cybernetics

160 00 Prague 6, Jugoslávských partyzánů 1580/3, Czech Republic

http://people.ciirc.cvut.cz/hlavac, vaclav.hlavac@cvut.cz

also Center for Machine Perception, http://cmp.felk.cvut.cz

Courtesy: M.I. Schlesinger, V. Franc.

Outline of the talk:

- A classifier, dichotomy, a multi-class classifier.
- A linear discriminant function.
- Learning a linear classifier.
- Perceptron and its learning.
- ε-solution.
- Learning for infinite training sets.

A classifier

Analyzed object is represented by

X – a space of observations, a vector space of dimension n.

Y – a set of hidden states.

The aim of the classification is to determine a relation between X and Y, i.e. to find a discriminant function $f\colon X \rightarrow Y$.

Classifier $q\colon X \rightarrow J$ maps observations $X^n \rightarrow$ set of class indices J, $J = 1, \ldots, |Y|$.

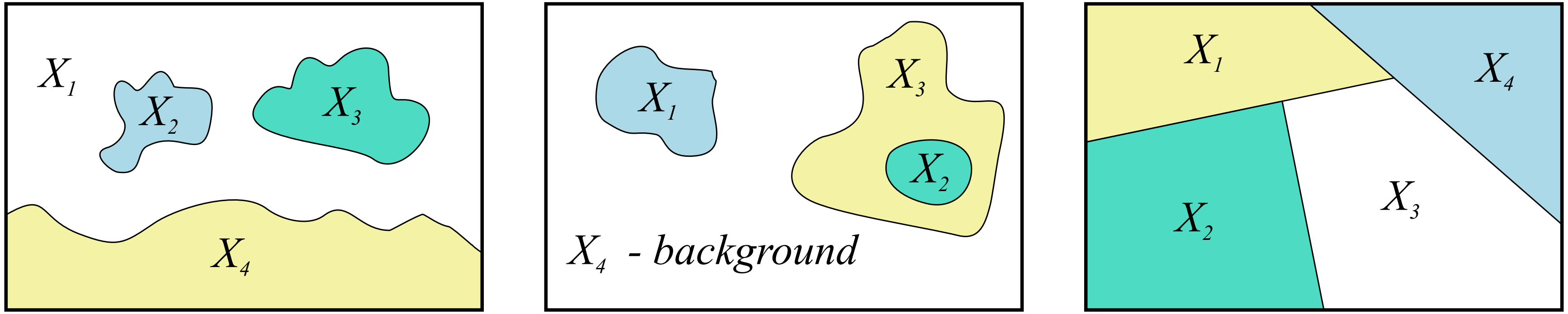
Mutual exclusion of classes is required

$X = X_1 \cup X_2 \cup \ldots \cup X_{|Y|}$,

$X_i \cap X_j = \emptyset$, $i \neq j$, $i, j = 1 \ldots |Y|$.

Classifier, an illustration

- A classifier partitions the observation space X into class-labelled regions X_i, $i = 1, \ldots, |Y|$.
- Classification determines, to which region X_i corresponding to a hidden state y_i an observed feature vector x belongs.
- Borders between regions are called decision boundaries.

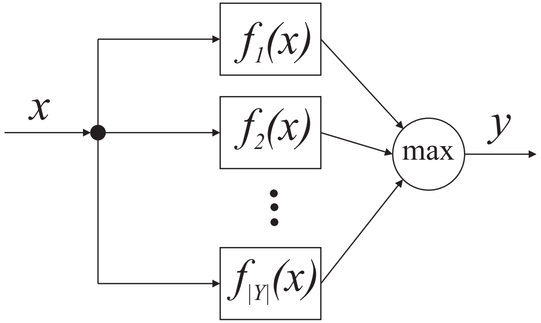


Several possible arrangements of decision boundaries corresponding to hidden states y_i (classes in a more specific case).

A multi-class decision strategy

- Discriminant functions $f_i(x)$ should have the (ideal) property:

$$f_i(x) > f_j(x) \text{ for } x \in \text{class } i\,,\ i \neq j.$$

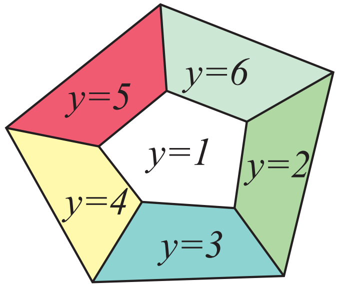


Strategy: $j = \operatorname*{argmax}_j f_j(x)$

- However, it is uneasy to find such a discriminant function. Most 'good classifiers' are dichotomic (as perceptron, SVM).
- The usual solution: One-against-All classifier, One-against-One classifier.

Linear discriminant function

- $f_j(x) = \langle w_j, x \rangle + b_j$, where $\langle\ \rangle$ denotes a scalar product.
- A strategy $j = \operatorname*{argmax}_j f_j(x)$ divides X into $|Y|$ convex regions.

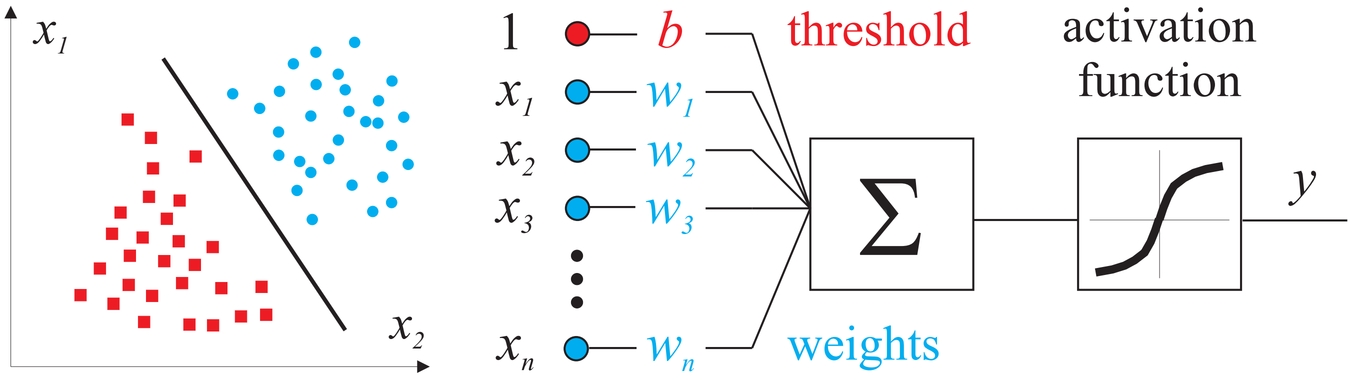


The classes separation corresponds to the Voronoi diagram (a term from computational geometry). The aim is to obtain a partition filling the space completely and reflecting the proximity information. Voronoi diagram is a partition of a space (plane in a special case) into regions close to each of given objects (seeds). The region corresponding to a particular seed consists of all points closer to that seed than to other seeds.

Dichotomy, two classes only

$|Y| = 2$, i.e. two hidden states (typically also classes)

$$q(x) = \begin{cases} y = 1\,, & \text{if} \quad \langle w, x\rangle + b \geq 0\,, \\ y = 2\,, & \text{if} \quad \langle w, x\rangle + b < 0\,. \end{cases}$$



Perceptron by F. Rosenblatt 1958

Learning linear classifiers

The aim of learning is to estimate classifier $q(x)$ parameters w_i, b_i for $\forall i$.

The learning algorithms differ by

- The character of training set
  1. Finite set consisting of individual observations and hidden states, i.e., $\{(x_1, y_1) \dots (x_L, y_L)\}$.
  2. Infinite sets described e.g. by a mixture of Gaussian distributions.
- Learning task formulations.

Learning tasks formulations for finite training sets

Empirical risk minimization:

- True risk is approximated by $R_{\text{emp}}(q(x,\Theta)) = \frac{1}{L}\sum_{i=1}^{L} W(q(x_i,\Theta), y_i)$, where W is a penalty.
- Learning is based on the empirical minimization principle $\Theta^* = \operatorname*{argmin}_{\Theta} R_{\text{emp}}(q(x,\Theta))$.
- Examples of learning algorithms: Perceptron, Back-propagation.

Structural risk minimization:

- True risk is approximated by a guaranteed risk (a regularizer securing upper bound of the risk is added to the empirical risk, Vapnik-Chervonenkis theory of learning).
- Example: Support Vector Machine (SVM).

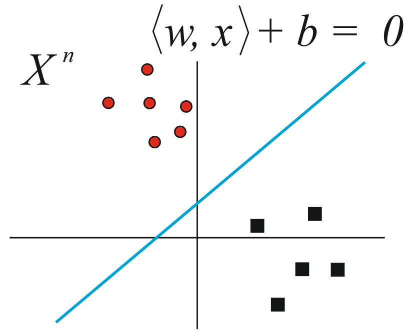
Perceptron, the history

- The perceptron (F. Rosenblatt, 1958) was intended to be a machine (even it was first implemented as a program on IBM 704).
- This machine was designed for image recognition: it had an array of 400 photocells, randomly connected to the “neurons”. Weights were encoded in potentiometers, and weight updates during learning were performed by electric motors
- Although the perceptron initially seemed promising, its limitations appeared. In 1969 a famous book entitled Perceptrons by Marvin Minsky and Seymour Papert showed that it was impossible for these classes of network to learn an XOR function.
- It is often believed (incorrectly) that they also conjectured that a similar result would hold for a multi-layer perceptron network.
- The often-miscited Minsky/Papert text caused a significant decline in interest and funding of neural network research.

Courtesy: Wikipedia

(A single-layer) perceptron learning; Task formulation

Input: $T = \{(x_1, y_1) \dots (x_L, y_L)\}$, $y_i \in \{1, 2\}$, $i = 1, \dots, L$, $\dim(x_i) = n$.



Output: a weight vector w, offset b for $\forall j \in \{1, \dots, L\}$ satisfying:

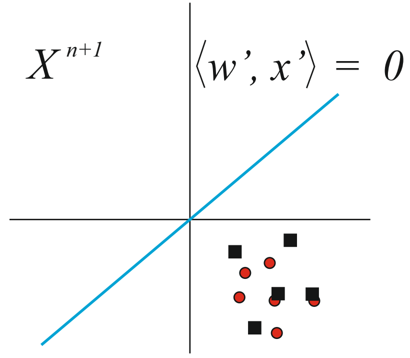
$$\langle w, x_j \rangle + b \geq 0 \text{ for } y = 1,$$

$$\langle w, x_j \rangle + b < 0 \text{ for } y = 2.$$

The task can be formally transcribed to a single inequality $\langle w', x'_j \rangle \geq 0$ by embedding it into $n + 1$ dimensional space, where $w' = [w \quad b]$,

$$x' = \begin{cases} [x \quad 1] \text{ for } y = 1\,, \\ -[x \quad 1] \text{ for } y = 2\,. \end{cases}$$

We drop the primes and go back to w, x notation.



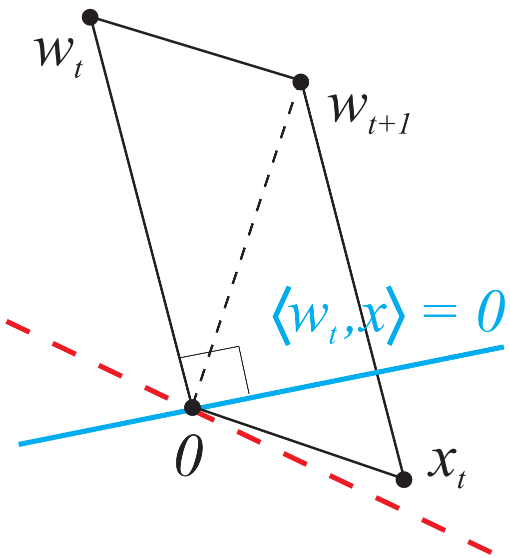
Perceptron learning: the algorithm 1957

Input: $T = \{(x_1, y_1) \ldots (x_L, y_L)\}$.
Output: a weight vector w.

The Perceptron algorithm

1. $w_1 = 0$.
2. A wrongly classified observation x_j is sought, i.e., $\langle w_t, x_j \rangle < 0$, $j \in \{1, \ldots, L\}$.
3. If there is no misclassified observation then the algorithm terminates otherwise
   $w_{t+1} = w_t + x_j$.
4. Goto 2.

F. Rosenblatt: The perceptron: a probabilistic model for information storage and organization in the brain,"Psychological review 65.6 (1958): 386

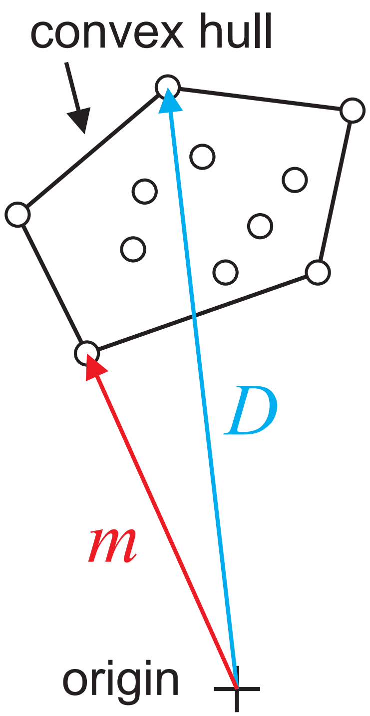


Perceptron update rule

The perceptron algorithm was invented at the Cornell Aeronautical Laboratory by Frank Rosenblatt (∗1928, †1971).

Novikoff theorem, 1962

- *Proves that the Perceptron algorithm converges in a finite number steps if the linearly separable solution exists.*
- Let $\overline{X}$ denotes a convex hull of points (set of observations) X.
- Let $D = \max_i |x_i|$, $m = \min_{x \in \overline{X}} |x_i| > 0$.



Novikoff theorem: If the data is linearly separable then there exists a number $t^* \leq \frac{D^2}{m^2}$, such that the vector w_{t^*} satisfies

$$\langle w_{t^*}, x_j \rangle > 0, \quad \forall j \in \{1, \ldots, L\}\,.$$

- *What if the data is not separable?*
- *How to terminate the perceptron learning?*

Idea of the Novikoff theorem proof

Let express bounds for $|w_t|^2$:

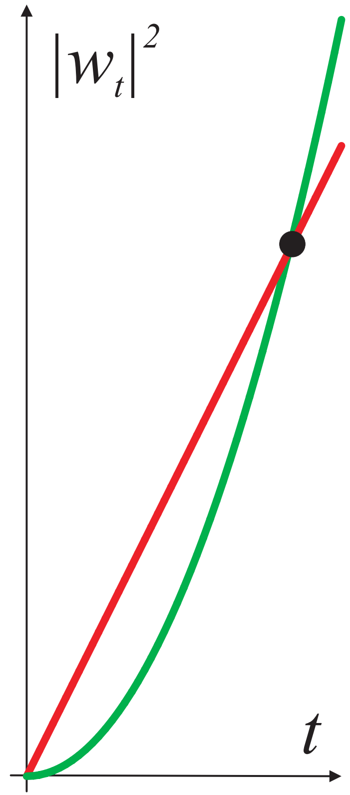
Upper bound:

$$\begin{aligned} |w_{t+1}|^2 &= |w_t + x_t|^2 = |w_t|^2 + 2\,\underbrace{\langle x_t, w_t \rangle}_{\leq 0} + |x_t|^2 \\ &\leq |w_t|^2 + |x_t|^2 \leq |w_t|^2 + D^2 \,. \end{aligned}$$

$|w_0|^2 = 0$, $|w_1|^2 \leq D^2$, $|w_2|^2 \leq 2D^2$, ..., $|w_{t+1}|^2 \leq t\,D^2$,
...

Lower bound: is given analogically $|w_{t+1}|^2 > t^2\,m^2$.

Solution: $t^2\,m^2 \leq t\,D^2 \Rightarrow t \leq \frac{D^2}{m^2}$.

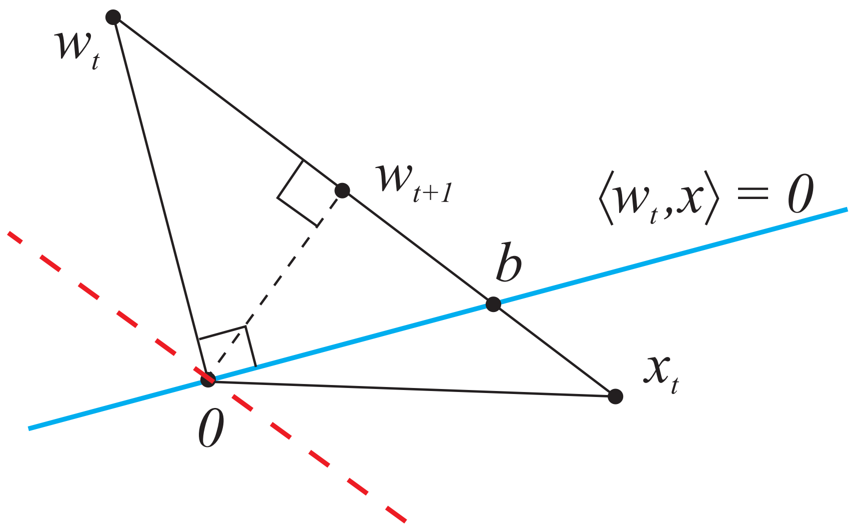


An alternative training algorithm Kozinec (1973)

Input: $T = \{(x_1, y_1) \ldots (x_L, y_L)\}$.

Output: a weight vector w^*.

1. $w_1 = x_j$, i.e., any observation.
2. A wrongly classified observation x_t is sought, i.e., $\langle w_t, x^j \rangle < 0$, $j \in J$.
3. If there is no wrongly classified observation then the algorithm finishes otherwise $w_{t+1} = (1-k) \cdot w_t + x_t \cdot k$, $k \in \mathbb{R}$, where $k = \underset{k}{\operatorname{argmin}} |(1-k) \cdot w_t + x_t \cdot k|$.
4. Goto 2.



Kozinec

Perceptron learning as an optimization problem (1)

Perceptron algorithm, the batch version, handling non-separability, another perspective:

- Input: $T = \{(x_1, y_1) \dots (x_L, y_L)\}$, $y_i \in \{1, 2\}$, $i = 1, \dots, L$, $\dim(x_i) = n$.
- Output: a weight vector w minimizing $J(w) = |\{x \in X : \langle w_t, x\rangle < 0\}|$ or, equivalently

$$J(w) = \sum_{x \in X:\ \langle w_t, x\rangle < 0} 1\,.$$

Q: Is the most common optimization method, the gradient descent $w_t = w - \eta \nabla J(w)$, applicable?

A: Unfortunately not. The gradient of $J(w)$ is either 0 or undefined.

Perceptron learning as an Optimization problem (2)

Let us redefine the cost function:

$$J_p(w) = \sum_{x \in X:\ \langle w, x \rangle < 0} \langle w, x \rangle \ .$$

$$\nabla J_p(w) = \frac{\partial J}{\partial w} = \sum_{x \in X:\ \langle w, x \rangle < 0} x \ .$$

- The Perceptron algorithm performs gradient descent optimization of $J_p(w)$.
- Learning by the empirical risk minimization is just an instance of an optimization problem.
- Either gradient minimization (backpropagation in neural networks) or convex (quadratic) minimization (called convex programming in mathematical literature) is used.

Perceptron algorithm

Classifier learning, a non-separable case, a batch version

Input: $T = \{(x_1, y_1) \dots (x_L, y_L)\}$, $y_i \in \{1, 2\}$, $i = 1, \dots, L$, $\dim(x_i) = n$.

Output: a weight vector w^*.

1. $w_1 = 0$, $E = |T| = L$, $w^* = 0$.
2. Find all misclassified observations $X^- = \{x \in X : \langle w_t, x \rangle < 0\}$.
3. If $|X^-| < E$ then $E = |X^-|$; $w^* = w_t$, the time of the last update $t_{lu} = t$.
4. If the termination condition $tc(w^*, t, t_{lu})$ then terminate else $w_{t+1} = w_t + \eta_t \sum_{x \in X^-} x$.
5. Goto 2.

- The algorithm converges with the probability 1 to the optimal solution.
- The convergence rate is not known.
- The termination condition $tc(.)$ is a complex function of the quality of the best solution, time since the last update $t - t_{lu}$ and requirements on the solution.

The optimal separating plane and the closest point to the convex hull

The problem of the optimal separation by a hyperplane

$$w^* = \underset{w}{\mathrm{argmax}} \min_j \left\langle \frac{w}{|w|}, x_j \right\rangle \tag{1}$$

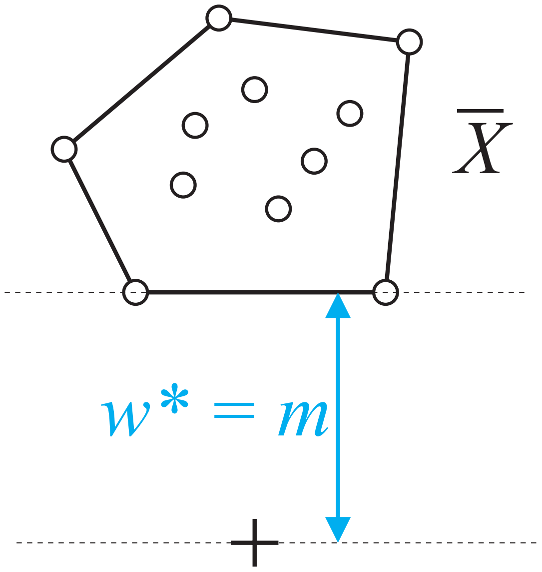
can be converted to a seek for the closest point to a convex hull (denoted by the overline) of the observations x in the training multi-set.

$$x^* = \underset{x \in \overline{X}}{\mathrm{argmin}} \; |x| \, .$$

It holds that x^* solves also the problem (1).

Recall that the classifier that maximizes the separation minimizes the structural risk R_{str}.

The convex hull, an illustration



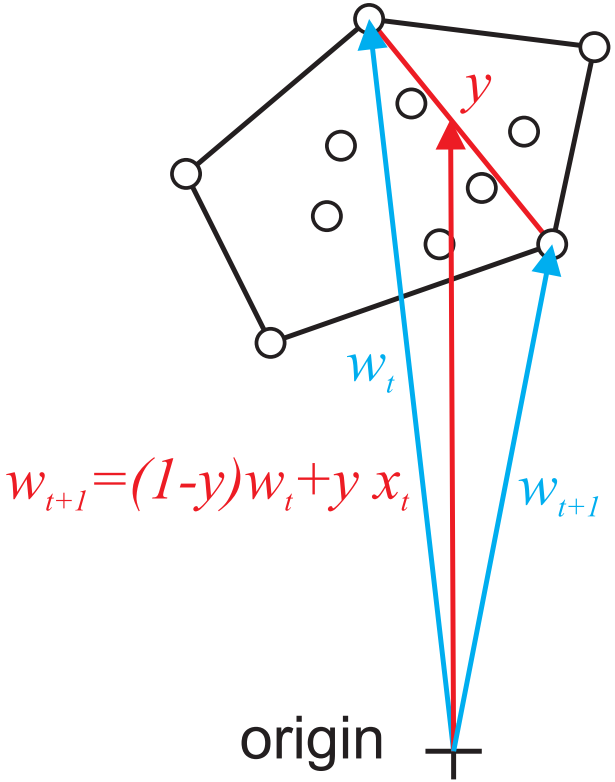
$$\min_j \left\langle \frac{w}{|w|}, x_j \right\rangle \leq m \leq |w| \, , \; w \in \overline{X} \, .$$

the lower bound the upper bound

ε-solution

- The aim is to speed up the algorithm.
- The allowed uncertainty ε is introduced.

$$|w^t| - \min_j \left\langle \frac{w^t}{|w^t|}, x_j \right\rangle \leq \varepsilon$$

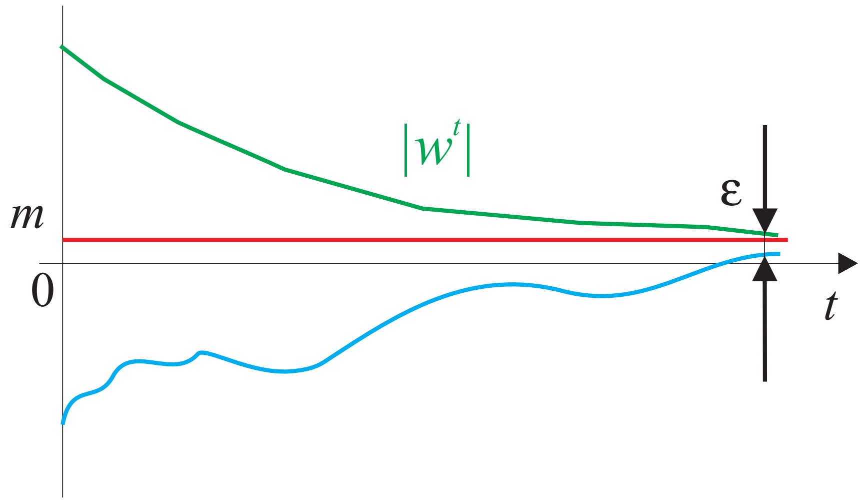


Kozinec and the ε-solution

The second step of Kozinec algorithm is modified to:

A wrongly classified observation x_t is sought, i.e.,

$$|w^t| - \min_j \left\langle \frac{w^t}{|w^t|}, x_t \right\rangle \geq \varepsilon$$



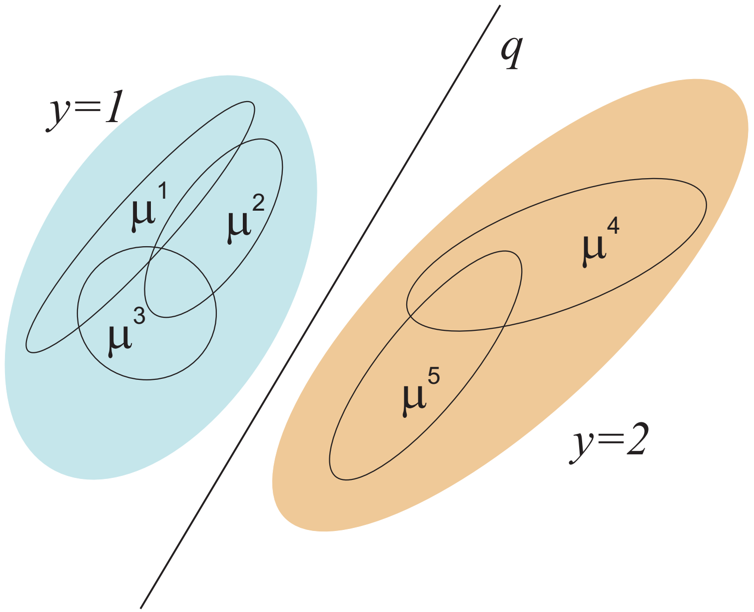
Learning task formulation for infinite training sets

The generalization of the Anderson's (1962) task by M.I. Schlesinger (1972) solves a quadratic optimization task.

- It solves the learning problem for a linear classifier and two hidden states only.
- It is assumed that a class-conditioned distribution $p_{X|Y}(x \mid y)$ corresponding to both hidden states are multi-dimensional Gaussian distributions.
- The mathematical expectation μ_y and the covariance matrix σ_y, $y = 1, 2$, of these probability distributions are not known.
- The Generalized Anderson task (abbreviated GAndersonT) is an extension of Anderson-Bahadur task (1962) which solved the problem when each of the two classes is modelled by a single Gaussian.

GAndersonT illustrated in the 2D space

Illustration of the statistical model, i.e., a mixture of Gaussians.



- The parameters of individual Gaussians μ_i, σ_i, $i = 1, 2, \ldots$ are known.
- Weights of the Gaussian components are unknown.