Introduction to Machine Learning with Applications

(A classification and engineering perspective)

Aggelos Pikrakis, Assist. Prof., Eng., Ph.D.

Dept. of Informatics, University of Piraeus, Greece

Office: 505

email: Pikrakis@unipi.gr

web: https://sites.google.com/site/aggelospikrakis/

Git: https://github.com/pikrakis

- This lecture was prepared based on material (slides and text) from the books:

  S. Theodorids and K. Koutroumbas, “Pattern Recognition, 4th Edition”, Academic Press, 2008

  S. Theodoridis, A. Pikrakis, K. Koutroumbas and D. Cavouras, “Introduction to Pattern Recognition: a Matlab Approach”, Academic Press, 2010

❖ Least Squares Methods

- If classes are NOT linearly separable, we shall compute the weights $w_1, w_2, \ldots, w_0$

  so that the difference between

  - The actual output of the classifier, $\underline{w}^T \underline{x}$, and
  - The desired outputs, e.g.

    $+1 \text{ if } \underline{x} \in \omega_1$

    $-1 \text{ if } \underline{x} \in \omega_2$

  to be SMALL

- SMALL, in the mean square error sense, means to choose $\underline{w}$ so that the cost function

  - $J(\underline{w}) \equiv E[(y - \underline{w}^T \underline{x})^2]$ is minimum
  - $\underline{\hat{w}} = \arg\min_{\underline{w}} J(\underline{w})$
  - y the corresponding desired responses

- Minimizing

$J(\underline{w})$ w.r.t. to $\underline{w}$ results in :

$$\frac{\partial J(\underline{w})}{\partial \underline{w}} = \frac{\partial}{\partial \underline{w}} E[(y - \underline{w}^T x)^2] = 0$$

$$= 2E[\underline{x}(y - \underline{x}^T \underline{w})] \Rightarrow$$

$$E[\underline{x}\underline{x}^T]\underline{w} = E[\underline{x}y] \Rightarrow$$

$$\boxed{\hat{\underline{w}} = R_x^{-1} E[\underline{x}y]}$$

where R_x is the autocorrelation matrix

$$R_x \equiv E[\underline{x}\underline{x}^T] = \begin{bmatrix} E[x_1 x_1] & E[x_1 x_2] \ldots & E[x_1 x_l] \\ \ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots \\ E[x_l x_1] & E[x_l x_2] \ldots & E[x_l x_l] \end{bmatrix}$$

and $E[\underline{x}y] = \begin{bmatrix} E[x_1 y] \\ \ldots \\ E[x_l y] \end{bmatrix}$ the crosscorrelation vector

➢ Multi-class generalization

- The goal is to compute M linear discriminant functions:

$$g_i(\underline{x}) = \underline{w}_i^T \underline{x}$$

according to the MSE.

- Adopt as desired responses y_i:

$$y_i = 1 \quad \text{if} \quad \underline{x} \in \omega_i$$

$$y_i = 0 \quad \text{otherwise}$$

- Let

$$\underline{y} = [y_1, y_2, \ldots, y_M]^T$$

- And the matrix

$$W = [\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_M]$$

- The goal is to compute W:

$$\hat{W} = \arg\min_{W} E\left[\left\| \underline{y} - W^T \underline{x} \right\|^2\right] = \arg\min_{W} E\left[\sum_{i=1}^{M} \left(y_i - \underline{w}_i^T \cdot \underline{x}\right)^2\right]$$

- The above is equivalent to a number M of MSE minimization problems. That is:

  Design each $\underline{w}_i$ so that its desired output is 1 for $\underline{x} \in \omega_i$ and 0 for any other class.

➢ Remark: The MSE criterion belongs to a more general class of cost function with the following important property:

- The value of $g_i(\underline{x})$ is an estimate, in the MSE sense, of the a-posteriori probability $P(\omega_i \mid \underline{x})$, provided that the desired responses used during training are $y_i = 1, \underline{x} \in \omega_i$ and 0 otherwise.

- **Mean square error regression**: Let $\underline{y} \in \Re^M$, $\underline{x} \in \Re^\ell$ be jointly distributed random vectors with a joint pdf $p(\underline{x}, \underline{y})$
  - The goal: Given the value of $\underline{x}$ estimate the value of $\underline{y}$. In the pattern recognition framework, given $\underline{x}$ one wants to estimate the respective label $y = \pm 1$.
  - The MSE estimate $\hat{\underline{y}}$ of $\underline{y}$ given $\underline{x}$ is defined as:

$$\hat{\underline{y}} = \arg\min_{\tilde{y}} E\left[\left\|y - \tilde{y}\right\|^2\right]$$

  - It turns out that:

$$\hat{\underline{y}} = E\left[\underline{y} \mid \underline{x}\right] \equiv \int_{-\infty}^{+\infty} \underline{y}\, p(\underline{y} \mid \underline{x})\, d\underline{y}$$

  The above is known as the regression of $\underline{y}$ given $\underline{x}$ and it is, in general, a non-linear function of $\underline{x}$. If $p(\underline{x}, \underline{y})$ is Gaussian the MSE regressor is linear.

❖ SMALL in the sum of error squares sense means

- $$J(\underline{w}) = \sum_{i=1}^{N} (y_i - \underline{w}^T \underline{x}_i)^2$$

  $(y_i, \underline{x}_i)$: training pairs that is, the input $\underline{x}_i$ and its corresponding class label y_i (±1).

- $$\frac{\partial J(\underline{w})}{\partial \underline{w}} = \frac{\partial}{\partial \underline{w}} \sum_{i=1}^{N} (y_i - \underline{w}^T \underline{x}_i)^2 = 0 \Rightarrow$$

$$\boxed{(\sum_{i=1}^{N} \underline{x}_i \underline{x}_i^T)\underline{w} = \sum_{i=1}^{N} \underline{x}_i y_i}$$

❖ Pseudoinverse Matrix

➢ Define

$$X = \begin{bmatrix} \underline{x}_1^T \\ \underline{x}_2^T \\ \dots \\ \underline{x}_N^T \end{bmatrix} \quad \text{(an } Nxl \text{ matrix)}$$

$$\underline{y} = \begin{bmatrix} y_1 \\ \dots \\ y_N \end{bmatrix} \text{ corresponding desired responses}$$

➢ $$X^T = [\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N] \quad \text{(an } lxN \text{ matrix)}$$

➢ $$X^T X = \sum_{i=1}^{N} \underline{x}_i \underline{x}_i^T$$

➢ $$X^T \underline{y} = \sum_{i=1}^{N} \underline{x}_i y_i$$

Thus $$(\sum_{i=1}^{N} \underline{x}_i^T \underline{x}_i)\underline{\hat{w}} = (\sum_{i=1}^{N} \underline{x}_i y_i)$$

$$(X^T X)\underline{\hat{w}} = X^T \underline{y} \Rightarrow$$

$$\underline{\hat{w}} = (X^T X)^{-1} X^T \underline{y}$$

$$= X^{\neq} \underline{y}$$

$$\boxed{X^{\neq} \equiv (X^T X)^{-1} X^T}$$ Pseudoinverse of X

- Assume $N=l \quad \Rightarrow \quad X$ square and invertible. Then

$$(X^T X)^{-1} X^T = X^{-1} X^{-T} X^T = X^{-1} \Rightarrow$$

$$\boxed{X^{\neq} = X^{-1}}$$

- Assume $N>l$. Then, in general, there is no solution to satisfy all equations simultaneously:

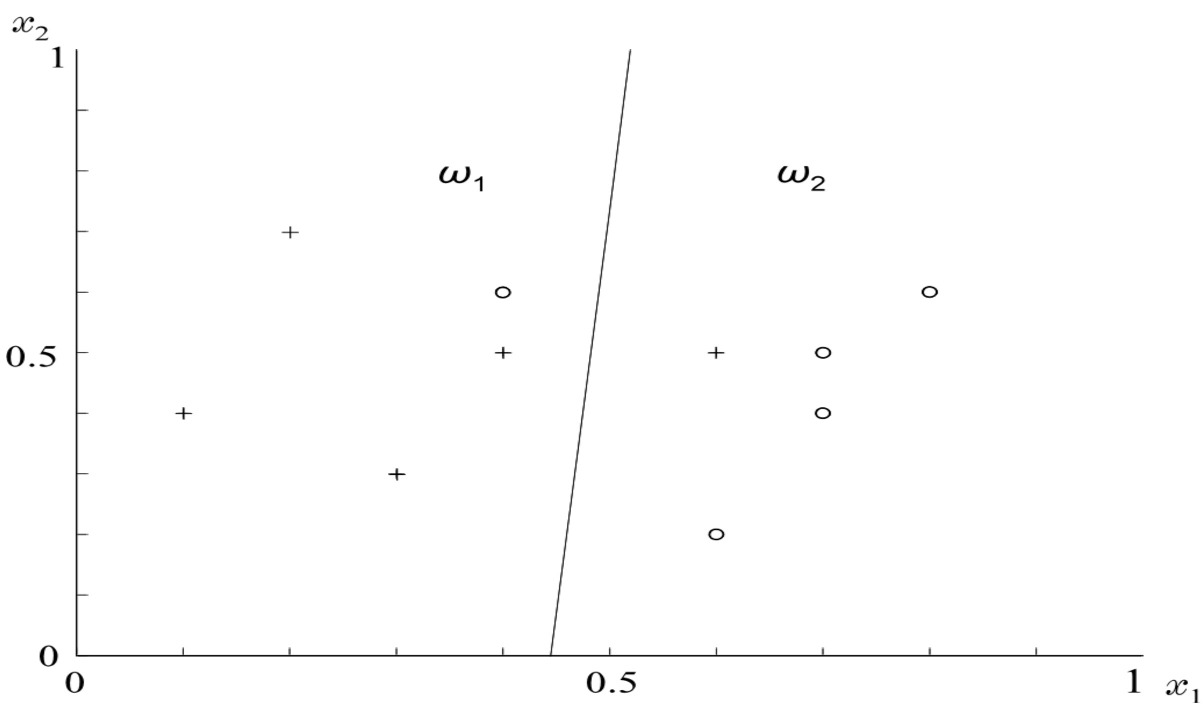
$$X\underline{w} = \underline{y}: \quad \begin{array}{c} \underline{x}_1^T \underline{w} = y_1 \\ \underline{x}_2^T \underline{w} = y_2 \\ \dots\dots \\ \underline{x}_N^T \underline{w} = y_N \end{array} \quad N \text{ equations} > l \text{ unknowns}$$

- The "solution" $\underline{w} = X^{\neq} \underline{y}$ corresponds to the minimum sum of squares solution

- Example:

$$\omega_1 : \begin{bmatrix}0.4\\0.5\end{bmatrix}, \begin{bmatrix}0.6\\0.5\end{bmatrix}, \begin{bmatrix}0.1\\0.4\end{bmatrix}, \begin{bmatrix}0.2\\0.7\end{bmatrix}, \begin{bmatrix}0.3\\0.3\end{bmatrix}$$

$$\omega_2 : \begin{bmatrix}0.4\\0.6\end{bmatrix}, \begin{bmatrix}0.6\\0.2\end{bmatrix}, \begin{bmatrix}0.7\\0.4\end{bmatrix}, \begin{bmatrix}0.8\\0.6\end{bmatrix}, \begin{bmatrix}0.7\\0.5\end{bmatrix}$$



$$X = \begin{bmatrix}0.4 & 0.5 & 1\\0.6 & 0.5 & 1\\0.1 & 0.4 & 1\\0.2 & 0.7 & 1\\0.3 & 0.3 & 1\\0.4 & 0.6 & 1\\0.6 & 0.2 & 1\\0.7 & 0.4 & 1\\0.8 & 0.6 & 1\\0.7 & 0.5 & 1\end{bmatrix} \begin{bmatrix}1\\1\\1\\1\\1\\-1\\-1\\-1\\-1\\-1\end{bmatrix} = \underline{y}$$

- $X^T X = \begin{bmatrix} 2.8 & 2.24 & 4.8 \\ 2.24 & 2.41 & 4.7 \\ 4.8 & 4.7 & 10 \end{bmatrix}, X^T \underline{y} = \begin{bmatrix} -1.6 \\ 0.1 \\ 0.0 \end{bmatrix}$

$$\underline{w} = (X^T X)^{-1} X^T \underline{y} = \begin{bmatrix} -3.13 \\ 0.24 \\ 1.34 \end{bmatrix}$$

❖ The Bias – Variance Dilemma

A classifier $g(\underline{x})$ is a learning machine that tries to predict the class label y of $\underline{x}$. In practice, a finite data set D is used for its training. Let us write $g(\underline{x}; D)$. Observe that:

- For some training sets, $D = \{(y_i, \underline{x}_i), i = 1, 2, ..., N\}$, the training may result to good estimates, for some others the result may be worse.

- The average performance of the classifier can be tested against the MSE optimal value, in the mean squares sense, that is:

$$E_D\left[\left(g(\underline{x}; D) - E[y \mid \underline{x}]\right)^2\right]$$

where E_D is the mean over all possible data sets D.

- The above is written as:

$$E_D\left[\left(g(\underline{x};D)-E[y\mid\underline{x}]\right)^2\right]=$$

$$\left(E_D\left[g(\underline{x};D)\right]-E[y\mid\underline{x}]\right)^2+E_D\left[\left(g(\underline{x};D)-E_D\left[g(\underline{x};D)\right]\right)^2\right]$$

- In the above, the first term is the contribution of the bias and the second term is the contribution of the variance.

- For a finite D, there is a trade-off between the two terms. Increasing bias it reduces variance and vice verse. This is known as the bias-variance dilemma.

- Using a complex model results in low-bias but a high variance, as one changes from one training set to another. Using a simple model results in high bias but low variance.

❖ LOGISTIC DISCRIMINATION

- Let an M-class task, $\omega_1, \omega_2, \ldots, \omega_M$. In logistic discrimination, the logarithm of the likelihood ratios are modeled via linear functions, i.e.,

$$\ln\left(\frac{P(\omega_i \mid \underline{x})}{P(\omega_M \mid \underline{x})}\right) = w_{i,0} + \underline{w}_i^{\,T}\,\underline{x}, \;\; i = 1,\ 2,\ \ldots,\ M\text{-}1$$

- Taking into account that

$$\sum_{i=1}^{M} P(\omega_i \mid \underline{x}) = 1$$

it can be easily shown that the above is equivalent with modeling posterior probabilities as:

$$P(\omega_M \mid \underline{x}) = \frac{1}{1 + \sum_{i=1}^{M-1} \exp\left(w_{i,0} + \underline{w}_i^{\ T} \underline{x}\right)}$$

$$P(\omega_i \mid \underline{x}) = \frac{\exp\left(w_{i,0} + \underline{w}_i^{\ T} \underline{x}\right)}{1 + \sum_{i=1}^{M-1} \exp\left(w_{i,0} + \underline{w}_i^{\ T} \underline{x}\right)}, \imath = 1,2,\ldots \mathrm{M} - 1$$

- For the two-class case it turns out that

$$P(\omega_2 \mid \underline{x}) = \frac{1}{1 + \exp\left(w_0 + \underline{w}^T \underline{x}\right)}$$

$$P(\omega_1 \mid \underline{x}) = \frac{\exp\left(w_0 + \underline{w}^T \underline{x}\right)}{1 + \exp\left(w_0 + \underline{w}^T \underline{x}\right)}$$

- The unknown parameters $\underline{w}_i, w_{i,0}, \; i = 1, 2, ..., M\text{-}1$ are usually estimated by maximum likelihood arguments.

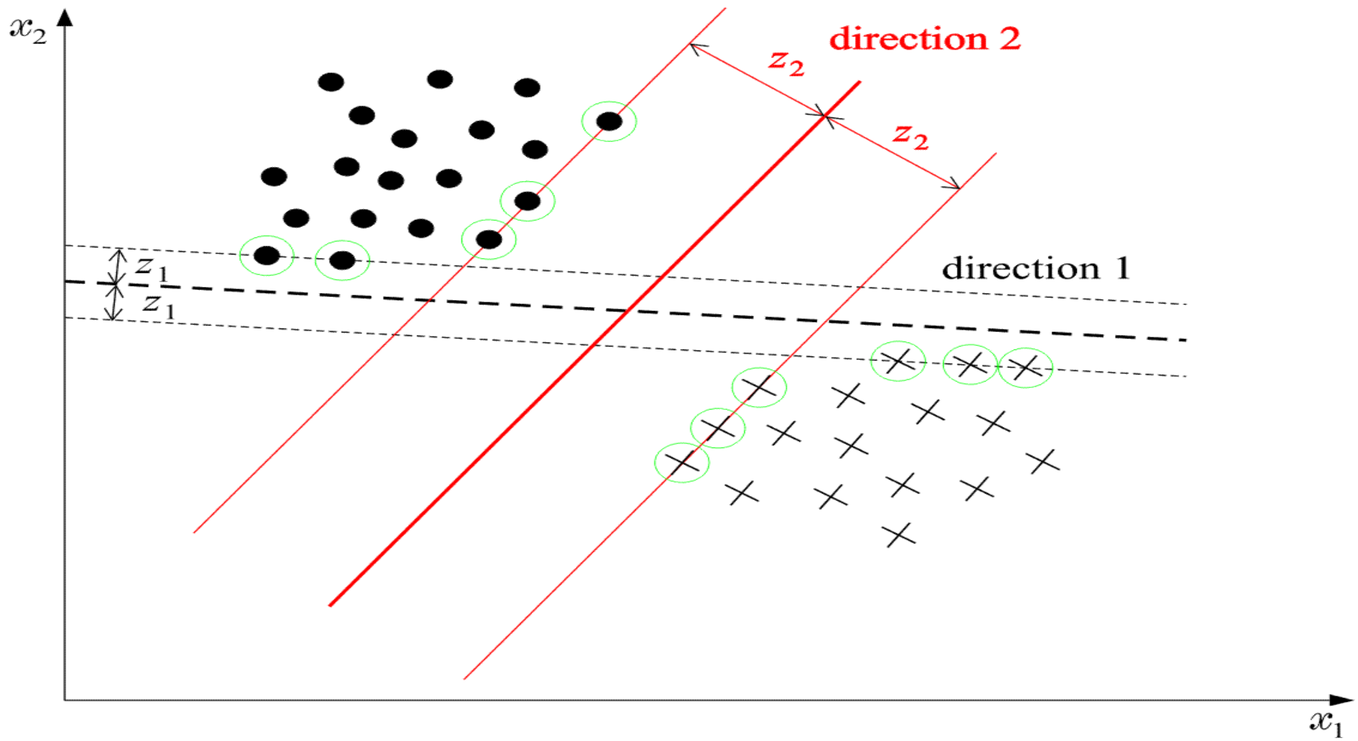
- Logistic discrimination is a useful tool, since it allows linear modeling and at the same time ensures posterior probabilities to add to one.

❖ Support Vector Machines

- The goal: Given two linearly separable classes, design the classifier

$$g(\underline{x}) = \underline{w}^T \underline{x} + w_0 = 0$$

that leaves the maximum margin from both classes



- Margin: Each hyperplane is characterized by
  - Its direction in space, i.e., $\underline{w}$
  - Its position in space, i.e., w_0
  - For EACH direction, $\underline{w}$, choose the hyperplane that leaves the SAME distance from the nearest points from each class. The margin is twice this distance.

- The distance of a point $\underline{\hat{x}}$ from a hyperplane is given by

$$z_{\hat{x}} = \frac{g(\underline{\hat{x}})}{\|\underline{w}\|}$$

- Scale, $\underline{w}, \underline{w}_0$, so that at the nearest points from each class the discriminant function is ±1:

$$|g(\underline{x})| = 1 \ \left\{ g(\underline{\mathrm{x}}) = +1 \text{ for } \omega_1 \text{ and } g(\underline{x}) = -1 \text{ for } \omega_2 \right\}$$

- Thus the margin is given by

$$\frac{1}{\|\underline{w}\|} + \frac{1}{\|\underline{w}\|} = \frac{2}{\|\underline{w}\|}$$

- Also, the following is valid

$$\underline{w}^T \underline{x} + w_0 \geq 1 \ \forall \underline{x} \in \omega_1$$

$$\underline{w}^T \underline{x} + w_0 \leq -1 \ \forall \underline{x} \in \omega_2$$

- SVM (linear) classifier

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

- Minimize

$$J(\underline{w}) = \frac{1}{2} \|\underline{w}\|^2$$

- Subject to

$$y_i(\underline{w}^T \underline{x}_i + w_0) \geq 1, \ i = 1,2,\ldots,N$$

$$y_i = 1, \text{for } \underline{x}_i \in \omega_i,$$

$$y_i = -1, \text{for } \underline{x}_i \in \omega_2$$

- The above is justified since by minimizing $\|\underline{w}\|$ the margin $\frac{2}{\|w\|}$ is maximised

- The above is a quadratic optimization task, subject to a set of linear inequality constraints. The Karush-Kuhh-Tucker conditions state that the minimizer satisfies:

  - (1) $\frac{\partial}{\partial \underline{w}} \mathrm{L}(\underline{w}, w_0, \underline{\lambda}) = \underline{0}$
  - (2) $\frac{\partial}{\partial w_0} L(\underline{w}, w_0, \underline{\lambda}) = 0$
  - (3) $\lambda_i \geq 0, i = 1,2,\ldots, N$
  - (4) $\lambda_i \left[y_i (\underline{w}^T \underline{x}_i + w_0) - 1 \right] = 0, i = 1,2,\ldots, N$
  - Where $L(\bullet,\bullet,\bullet)$ is the Lagrangian

$$L(\underline{w}, w_0, \underline{\lambda}) \equiv \frac{1}{2} \underline{w}^T \underline{w} - \sum_{i=1}^{N} \lambda_i [y_i (\underline{w}^T \underline{x}_i + w_0)]$$

- The solution: from the above, it turns out that

- $$\underline{w} = \sum_{i=1}^{N} \lambda_i y_i \underline{x}_i$$

- $$\sum_{i=1}^{N} \lambda_i y_i = 0$$

➢ Remarks:

- The Lagrange multipliers can be either zero or positive. Thus,

  – $$\underline{w} = \sum_{i=1}^{N_s} \lambda_i y_i \underline{x}_i$$

  where $N_s \leq N_0$, corresponding to positive Lagrange multipliers

  – From constraint (4) above, i.e.,

  $$\lambda_i [y_i (\underline{w}^T \underline{x}_i + w_0) - 1] = 0, \quad i = 1,2,\ldots,N$$

  the vectors contributing to $\underline{w}$ satisfy

  $$\underline{w}^T \underline{x}_i + w_0 = \pm 1$$

- These vectors are known as SUPPORT VECTORS and are the closest vectors, from each class, to the classifier.

- Once $\underline{w}$ is computed, w_0 is determined from conditions (4).

- The optimal hyperplane classifier of a support vector machine is UNIQUE.

- Although the solution is unique, the resulting Lagrange multipliers are not unique.

Dual Problem Formulation

- The SVM formulation is a convex programming problem, with
  - Convex cost function
  - Convex region of feasible solutions
- Thus, its solution can be achieved by its dual problem, i.e.,
  - maximize $\underset{\underline{\lambda}}{} \quad L(\underline{w}, w_0, \underline{\lambda})$
  - subject to

$$\underline{w} = \sum_{i=1}^{N} \lambda_i y_i \underline{x}_i$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\underline{\lambda} \geq \underline{0}$$

- Combine the above to obtain
  - maximize $\underset{\underline{\lambda}}{\text{maximize}}$

$$\left(\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{ij}\lambda_i\lambda_j y_i y_j \underline{x}_i^T \underline{x}_j\right)$$

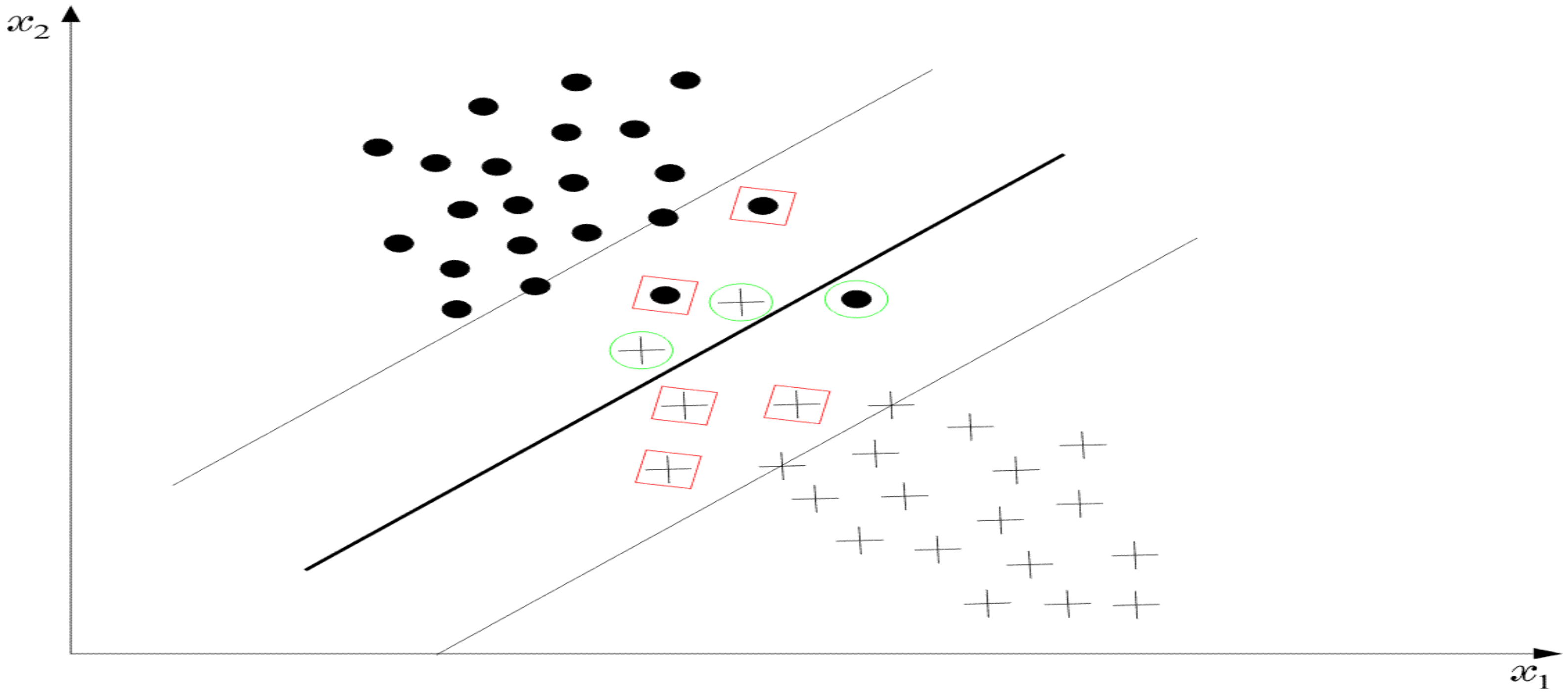
  - subject to

$$\sum_{i=1}^{N}\lambda_i y_i = 0$$

$$\underline{\lambda} \geq \underline{0}$$

- Remarks:
  - Support vectors enter via inner products

- Non-Separable classes



In this case, there is no hyperplane such that

$$\underline{w}^T \underline{x} + w_0 (><) 1, \ \forall \underline{x}$$

- Recall that the margin is defined as twice the distance between the following two hyperplanes

$$\underline{w}^T \underline{x} + w_0 = 1$$

and

$$\underline{w}^T \underline{x} + w_0 = -1$$

- The training vectors belong to <u>one</u> of <u>three</u> possible categories

1) Vectors outside the band which are correctly classified, i.e.,

$$y_i(\underline{w}^T \underline{x} + w_0) > 1$$

2) Vectors inside the band, and correctly classified, i.e.,

$$0 \leq y_i(\underline{w}^T \underline{x} + w_0) < 1$$

3) Vectors misclassified, i.e.,

$$y_i(\underline{w}^T \underline{x} + w_0) < 0$$

- All three cases above can be represented as

$$y_i(\underline{w}^T \underline{x} + w_0) \geq 1 - \xi_i$$

1) $\rightarrow \xi_i = 0$
2) $\rightarrow 0 < \xi_i \leq 1$
3) $\rightarrow 1 < \xi_i$

ξ_i are known as slack variables

- The goal of the optimization is now two-fold
  - Maximize margin
  - Minimize the number of patterns with $\xi_i > 0$,
    One way to achieve this goal is via the cost

$$J(\underline{w}, w_0, \underline{\xi}) = \frac{1}{2}\|\underline{w}\|^2 + C\sum_{i=1}^{N} I(\xi_i)$$

where C is a constant and

$$I(\xi_i) = \begin{Bmatrix} 1 & \xi_i > 0 \\ 0 & \xi_i = 0 \end{Bmatrix}$$

  - $I(.)$ is not differentiable. In practice, we use an approximation
  - $J(\underline{w}, w_0, \underline{\xi}) = \frac{1}{2}\|\underline{w}\|^2 + C\sum_{i=1}^{N} \xi_i$
  - Following a similar procedure as before we obtain

➢ KKT conditions

$$(1)\quad \underline{w} = \sum_{i=1}^{N} \lambda_i y_i \underline{x}_i$$

$$(2)\quad \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$(3)\quad C - \mu_i - \lambda_i = 0, i = 1,2,\ldots,N$$

$$(4)\quad \lambda_i [y_i (\underline{w}^T \underline{x}_i + w_0) - 1 + \xi_i] = 0, \quad i = 1,2,\ldots,N$$

$$(5)\quad \mu_i \xi_i = 0, \quad i = 1,2,\ldots,N$$

$$(6)\quad \mu_i, \lambda_i \geq 0, \quad i = 1,2,\ldots,N$$

- The associated dual problem

Maximize $$\max_{\underline{\lambda}} \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \underline{x}_i^T \underline{x}_j \right)$$

subject to

$$0 \leq \lambda_i \leq C, \; i = 1,2,\ldots,N$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0$$

- Remarks: The only difference with the separable class case is the existence of C in the constraints

- Training the SVM

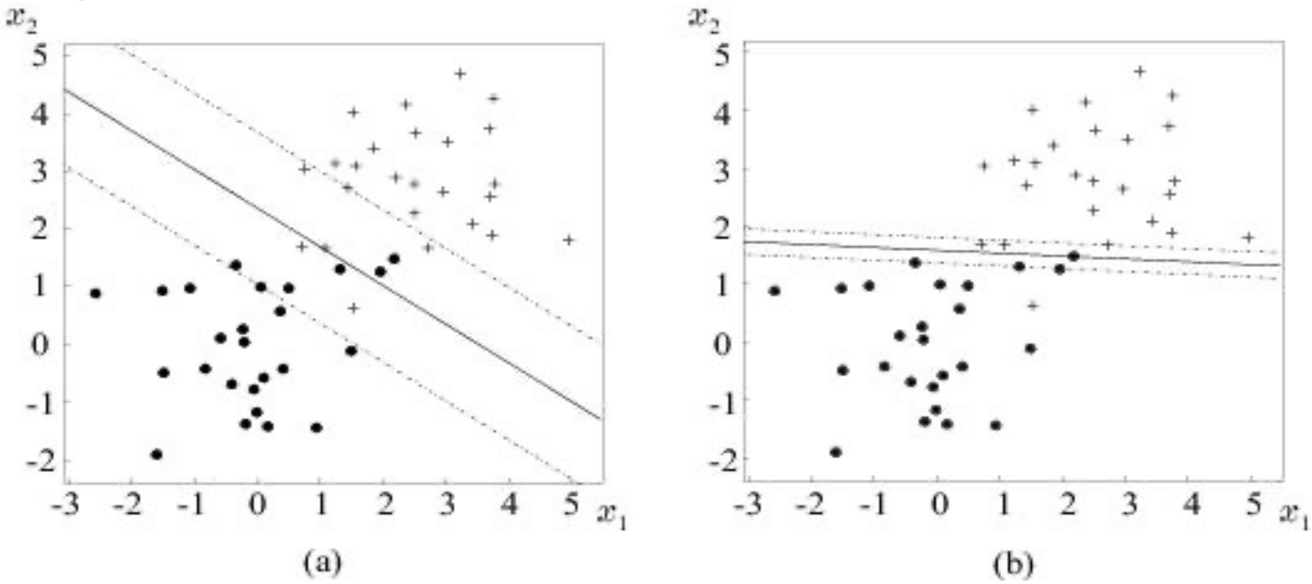
A major problem is the high computational cost. To this end, decomposition techniques are used. The rationale behind them consists of the following:

- Start with an arbitrary data subset (working set) that can fit in the memory. Perform optimization, via a general purpose optimizer.
- Resulting support vectors remain in the working set, while others are replaced by new ones (outside the set) that violate severely the KKT conditions.
- Repeat the procedure.
- The above procedure guarantees that the cost function decreases.
- Platt's SMO algorithm chooses a working set of two samples, thus analytic optimization solution can be obtained.

- Multi-class generalization

  Although theoretical generalizations exist, the most popular in practice is to look at the problem as M two-class problems (one against all).

- Example:



- Observe the effect of different values of C in the case of non-separable classes.