PDEEC – Machine Learning 2018/19

Lecture - Introduction to Support Vector Machines

Jaime S. Cardoso
jaime.cardoso@inesctec.pt

INESC TEC and Faculdade Engenharia, Universidade do Porto

Nov. 15, 2018

Background

Convex Hull

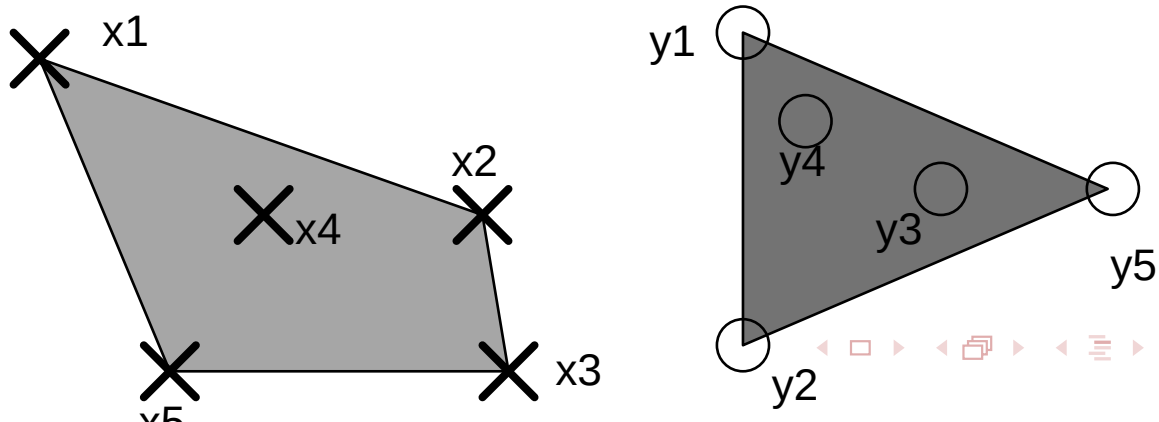
- Convex Hull
  - Given a set of points $\{\mathbf{x}_i\}$ we define the convex hull to be the set of all points $\mathbf{x}$ given by

$$\mathbf{x} = \sum_i \alpha_i \mathbf{x}_i$$

$$\text{subject to:} \quad \alpha_i \geq 0$$

$$\sum_i \alpha_i = 1$$

  - Given two sets of points $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_i\}$ if their convex hull intersect, the two sets of points cannot be linearly separable
  - Given two sets of points $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_i\}$ if they are linearly separable, their convex hull do not intersect



Background

Convex Hull

- Reduced Convex Hulls
  - Given a set of points $\{\mathbf{x}_i\}$ we define the *reduced* convex hull to be the set of all points $\mathbf{x}$ given by

$$\mathbf{x} = \sum_i \alpha_i \mathbf{x}_i$$

subject to:

$$\alpha_i \geq 0$$

$$\textstyle\sum_i \alpha_i = 1$$

$$\alpha_i \leq \alpha, \quad \tfrac{1}{N} \leq \alpha \leq 1$$

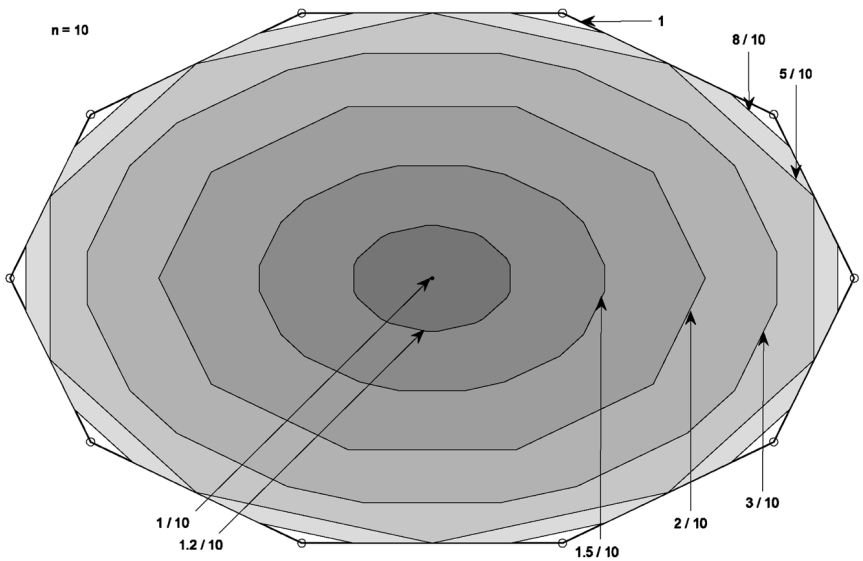


Figure: from [4]

Background

Distance of the origin to a plane.

- plane defined by equation $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$
  - $y(\mathbf{x}_A) = 0$ if $\mathbf{x}_A \in$ to plane
  - $\mathbf{w}$ is orthogonal to every vector lying on the plane: $\mathbf{w}^T(\mathbf{x}_A - \mathbf{x}_B) = 0$, with $\mathbf{x}_A$ and $\mathbf{x}_B$ on the plane
  - the distance of the plane to the origin is $| < \mathbf{x}, \frac{\mathbf{w}}{||\mathbf{w}||} > |$ (projection of any point of the plane in the direction of $\mathbf{w}$) $= \frac{|\mathbf{w}^T\mathbf{x}|}{||\mathbf{w}||} = \frac{|b|}{||\mathbf{w}||}$

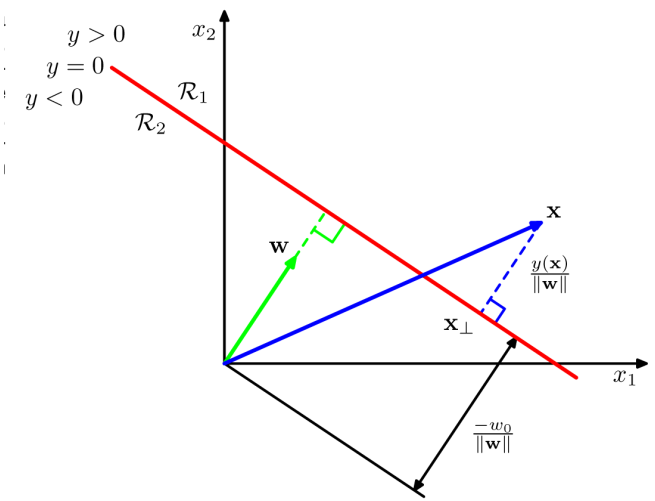


Figure: from [1]

Background

Distance of a point P to a plane (I).

- 1st proof
  - by translation, make P the origin $\mathcal{O}$ of a new system of coordinates: $\mathbf{x}' = \mathbf{x} - P$
  - the equation of the plane $\mathbf{w}^T\mathbf{x} + b = 0$ becomes $\mathbf{w}^T(\mathbf{x}' + P) + b = 0 \leftrightarrow \mathbf{w}^T\mathbf{x}' + (\mathbf{w}^T P + b) = 0$
  - the distance of the origin to the plane is $= \frac{|(\mathbf{w}^T P + b)|}{||\mathbf{w}||} = \frac{|y(P)|}{||\mathbf{w}||}$
- 2nd proof
  - $P = P_\perp + P_{//} = P_\perp + r\frac{\mathbf{w}}{||\mathbf{w}||} \Rightarrow \mathbf{w}^T P + b = \mathbf{w}^T P_\perp + b + \mathbf{w}^T r\frac{\mathbf{w}}{||\mathbf{w}||} = 0 + r||\mathbf{w}|| \Rightarrow r = \frac{\mathbf{w}^T P + b}{||\mathbf{w}||} = \frac{y(P)}{||\mathbf{w}||} \Rightarrow |r| = \frac{|y(P)|}{||\mathbf{w}||}$

Background

Distance of a point P to a plane (II)

- 3rd proof
  - the problem of finding the minimum of $||(\mathbf{x}-P)||$ is equivalent to finding the minimum of $||(\mathbf{x}-P)||^2 = (\mathbf{x}-P)^T(\mathbf{x}-P)$
  - using Lagrange multipliers, the problem of finding the minimum of $(\mathbf{x}-P)^T(\mathbf{x}-P)$ subject to $\mathbf{w}^T\mathbf{x}+b=0$ is equivalent to the problem of finding the minimum of $\mathcal{L} = (\mathbf{x}-P)^T(\mathbf{x}-P) + \lambda(\mathbf{w}^T\mathbf{x}+b)$, for the variables $\mathbf{x}$ and λ. $\begin{cases} \frac{\partial \mathcal{L}}{\partial \mathbf{x}} = 2(\mathbf{x}-P) + \lambda\mathbf{w} = 0 \\ \frac{\partial \mathcal{L}}{\partial \lambda} = \mathbf{w}^T\mathbf{x}+b = 0 \end{cases}$
  - multiplying the 1st equation by $\mathbf{w}^T$ and using the 2nd for $\mathbf{w}^T\mathbf{x} = -b$, we obtain $\lambda = \frac{-2(\mathbf{w}^T P+b)}{\mathbf{w}^T\mathbf{w}}$
  - using this value of λ on the 1st equation we get $\mathbf{x}-P = \frac{(\mathbf{w}^T P+b)\mathbf{w}}{\mathbf{w}^T\mathbf{w}}$. The norm of this vector comes as $||\mathbf{x}-P|| = \frac{|\mathbf{w}^T P+b|\ ||\mathbf{w}||}{\mathbf{w}^T\mathbf{w}} = \frac{|\mathbf{w}^T P+b|}{||\mathbf{w}||} = \frac{|y(P)|}{||\mathbf{w}||}$
- algebric distance of a point P to a plane: $y(P)/||\mathbf{w}||$

Background

Lagrange Multipliers (I)

Maximizing f($\mathbf{x}$) subject to $g_i(\mathbf{x}) = 0$

- Consider the problem of finding the maximum of a function $f(\mathbf{x}), \mathbf{x} \in R^d$, subject to a set of constraints $g_i(\mathbf{x}) = 0, i = 1..m$
- There exists a $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_m)^T$, such that on the maximum $\mathbf{x}_0$
  $\nabla f(\mathbf{x}_0) = \lambda_1 \nabla g_1(\mathbf{x}_0) + \lambda_2 \nabla g_2(\mathbf{x}_0) + \cdots + \lambda_m \nabla g_m(\mathbf{x}_0)$

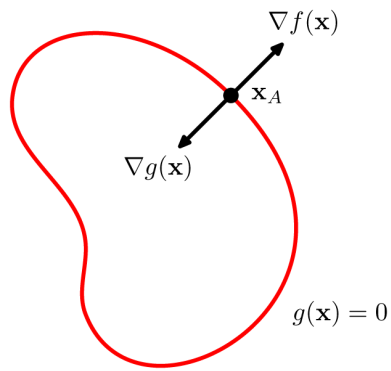


Figure: from[1]

- $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_m)$ are called the lagrange multipliers
- Introducing the Lagrangian function defined by $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_i \lambda_i g_i(\mathbf{x})$, we see that
  - $\partial L / \partial \lambda_i = g_i(\mathbf{x})$
  - $\nabla L_{\mathbf{x}} = \nabla f(\mathbf{x}) + \lambda_1 \nabla g_1(\mathbf{x}) + \lambda_2 \nabla g_2(\mathbf{x}) + \cdots + \lambda_m \nabla g_m(\mathbf{x})$
  - Just find the stationary point of $L(x, \lambda)$ with respect to both $\mathbf{x}$ and λ.

Background

Lagrange Multipliers (II)

Maximizing $f(\mathbf{x})$ subject to $g_i(\mathbf{x}) \geq 0$

- Two kinds of solutions:
  - the solution lies in $g(\mathbf{x}) > 0$, in which case the constraint is not active ($\lambda = 0$)
  - the solution lies in $g(\mathbf{x}) = 0$, in which case the constraint is active ($\lambda \neq 0$). NOTE that now $\lambda > 0$.

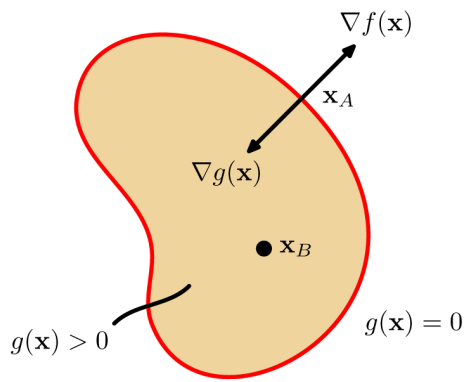


Figure: from [1]

Background

Lagrange Multipliers (III)

Karush-Kuhn-Tucker (KKT) conditions (it is a generalization of the method of Lagrange multipliers to inequality constraints)

- the solution to following nonlinear optimization problem

$$\begin{aligned} \text{argmax}_{\mathbf{x}} \quad & f(\mathbf{x}) \\ s.t. \quad & g_i(\mathbf{x}) \geq 0 \\ & h_i(\mathbf{x}) = 0 \end{aligned}$$

is obtained by optimizing

$$L(\mathbf{x}, \lambda_i, \nu_j) = f(\mathbf{x}) + \sum_i \lambda_i g_i(\mathbf{x}) + \sum_j \nu_j h_j(\mathbf{x})$$

with respect to $\mathbf{x}$ and λ subject to

- $h_j(\mathbf{x}) = 0$
- $g_i(\mathbf{x}) \geq 0$
- $\lambda_i \geq 0$
- $\lambda_i g_i(\mathbf{x}) = 0$

Background

Lagrange Multipliers (IV)

Karush-Kuhn-Tucker (KKT) conditions (it is a generalization of the method of Lagrange multipliers to inequality constraints)

- the solution to following nonlinear optimization problem

$$\begin{aligned} \text{argmin}_{\mathbf{x}} \quad & f(\mathbf{x}) \\ s.t. \quad & g_i(\mathbf{x}) \geq 0 \\ & h_i(\mathbf{x}) = 0 \end{aligned}$$

is obtained by optimizing

$$L(\mathbf{x}, \lambda_i, \nu_j) = f(\mathbf{x}) - \sum_i \lambda_i g_i(\mathbf{x}) - \sum_j \nu_j h_j(\mathbf{x})$$

with respect to $\mathbf{x}$ and λ subject to

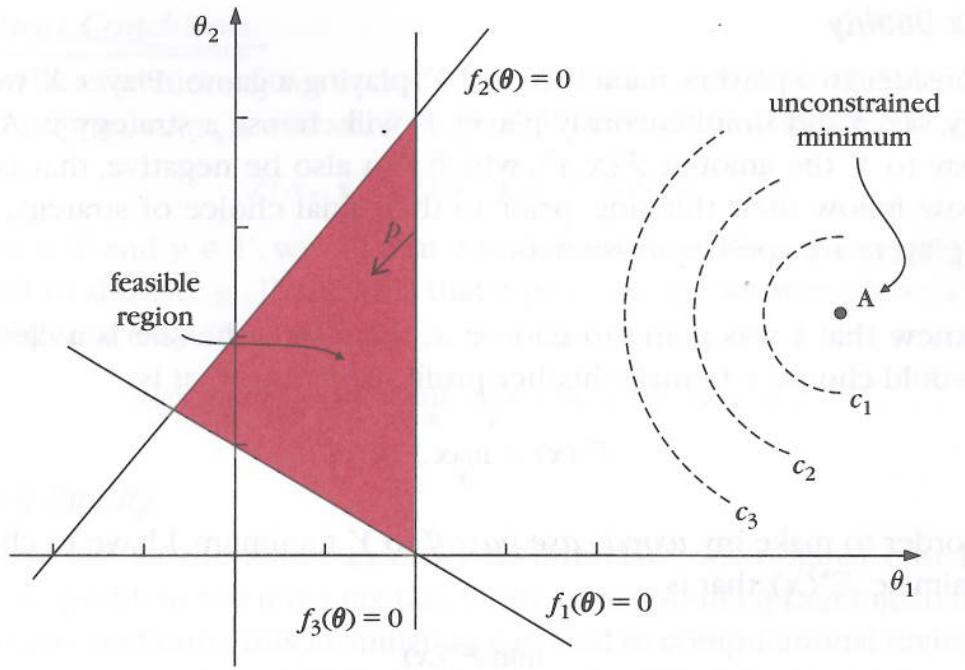
- $h_j(\mathbf{x}) = 0$
- $g_i(\mathbf{x}) \geq 0$
- $\lambda_i \geq 0$
- $\lambda_i g_i(\mathbf{x}) = 0$

Background

Lagrange Multipliers (V)

Constraints

$$f_1(\theta) = \theta_1 + 2\theta_2 - 2 \geq 0$$
$$f_2(\theta) = \theta_1 - \theta_2 + 2 \geq 0$$
$$f_3(\theta) = -\theta_1 + 2 \geq 0$$



- a constraint is called inactive if the corresponding Lagrange multiplier is zero
- a constraint f_i is called active if the corresponding Lagrange multiplier is nonzero (for which $f_i(\theta^*) = 0$).

Background

Min-Max Duality

- Primal:

$$\min_{\mathbf{x}} \max_{\mathbf{y}} \mathcal{F}(\mathbf{x}, \mathbf{y})$$

- Dual:

$$\max_{\mathbf{y}} \min_{\mathbf{x}} \mathcal{F}(\mathbf{x}, \mathbf{y})$$

-

$$\max_{\mathbf{y}} \min_{\mathbf{x}} \mathcal{F}(\mathbf{x}, \mathbf{y}) \leq \min_{\mathbf{x}} \max_{\mathbf{y}} \mathcal{F}(\mathbf{x}, \mathbf{y})$$

Background

Lagrangian Duality (I)

- $$\begin{aligned} \text{minimize}_{\mathbf{x}} \quad & J(\mathbf{x}) \\ s.t. \quad & g_i(\mathbf{x}) \geq 0 \end{aligned}$$

- $$L(\mathbf{x}, \lambda_i) = J(\mathbf{x}) - \sum_i \lambda_i g_i(\mathbf{x})$$

- since $\lambda_i \geq 0$ and $g_i(\mathbf{x}) \geq 0$ then

$$J(\mathbf{x}) = \max_{\lambda} L(\mathbf{x}, \lambda_i)$$

- the original problem is equivalent to

$$\min_{\mathbf{x}} \max_{\lambda \geq 0} L(\mathbf{x}, \lambda)$$

Convex Programming

- Assume that $J(\mathbf{x})$ is convex
- Assume that g_i are convex
- then

$$\min_{\mathbf{x}} \max_{\lambda \geq 0} L(\mathbf{x}, \lambda) =$$

$$= \max_{\lambda \geq 0} \min_{\mathbf{x}} L(\mathbf{x}, \lambda)$$

Wolfe Dual Representation

- A convex programming problem is equivalent to

$$\max_{\lambda \geq 0} L(\mathbf{x}, \lambda)$$

subject to

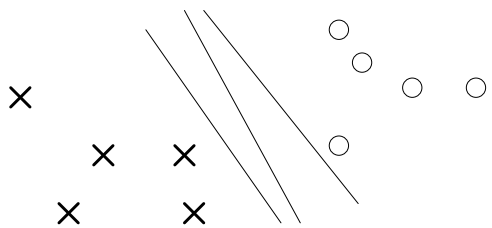
$$\frac{\partial}{\partial \mathbf{x}} L(\mathbf{x}, \lambda) = 0$$

Support vector machines

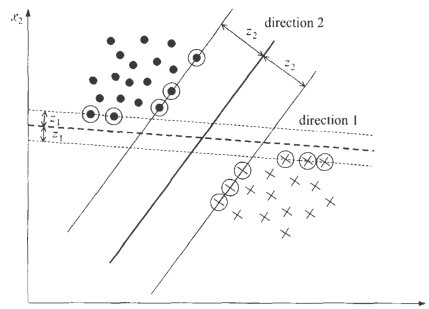
Linearly separable classes (I)

The training set comprises N input vectors $\mathbf{x}_1, \cdots, \mathbf{x}_N$, with corresponding target values $y_1, \cdots, y_N$, where $y_i \in \{-1, 1\}$. New data points $\mathbf{x}$ will be classified according the sign of the model $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$

- *margin* - smallest distance between the decision boundary and any of the samples
- In SVMs the decision boundary is chosen to be the one for which the margin is maximized



(a) An example of a linearly separable two class problem with multiple possible linear classifiers.



(b) SVM option: maximize the margin (from [2])

Support vector machines

Linearly separable classes (II)

- the distance of a training point $\mathbf{x}_n$ (correctly classified) to the decision surface is given by $\frac{y_n y(\mathbf{x}_n)}{||\mathbf{w}||} = \frac{y_n(\mathbf{w}^T\mathbf{x}+b)}{||\mathbf{w}||}$
- thus, we want

$$\text{argmax}_{\mathbf{w},b}\left\{\frac{1}{||\mathbf{w}||}\min(y_n(\mathbf{w}^T\mathbf{x}_n+b))\right\}$$

- note now that the solution to this problem is not completely well-defined: if $(\mathbf{w}, b)$ is a solution to this optimization problem, so it will be any solution of the form $(c\mathbf{w}, cb)$, where c is a constant
- we can impose an additional constraint to completely define the problem. The constraint can be chosen to simplify the formulation.

Support vector machines

Linearly separable classes (III)

- set $(y_n(\mathbf{w}^T\mathbf{x}_n + b)) = 1$ for the point closest to the surface. In this case all points should satisfy $(y_n(\mathbf{w}^T\mathbf{x}_n + b)) \geq 1, \quad n = 1, \cdots, N$
- We obtain the canonical representation:

$$\text{argmax}_{\mathbf{w},b} \frac{1}{||\mathbf{w}||}$$

subject to: $(y_n(\mathbf{w}^T\mathbf{x}_n + b)) \geq 1, \quad n = 1, \cdots, N$

- equivalently:

$$\text{argmin}_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2$$

subject to: $(y_n(\mathbf{w}^T\mathbf{x}_n + b)) \geq 1, \quad n = 1, \cdots, N$

- this is the **primal formulation** for the linearly separable case

Support vector machines

Linearly separable classes (IV)

Using the lagrange multipliers and the Karush-Kuhn-Tucker (KKT) conditions, we obtain the **dual formulation**:

- $L(\mathbf{w}, b, \lambda) = \frac{1}{2}||\mathbf{w}||^2 - \sum_n \lambda_n \{y_n(\mathbf{w}^T\mathbf{x}_n + b) - 1\}$
-

$$\partial L/\partial \mathbf{w} = 0 \Leftrightarrow \mathbf{w} - \sum_n \lambda_n y_n \mathbf{x}_n = 0$$

$$\partial L/\partial b = 0 \Leftrightarrow \sum_n \lambda_n y_n = 0$$

$$\lambda_i \geq 0, i = 1, \cdots, N$$

$$\lambda_i(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) = 0, i = 1, \cdots, N$$

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) \geq 0, i = 1, \cdots, N$$

Support vector machines

Linearly separable classes (V)

After a bit of algebra we end up with the equivalent optimization task, called the dual representation:

- $$\begin{aligned} \text{argmax}_\lambda \quad & \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to:} \quad & \sum_n \lambda_n y_n = 0 \\ & \lambda_i \geq 0, i = 1, \cdots, N \end{aligned}$$

- note that that now we have N variables, instead of the initial D variables (D is the dimension of the data space)
- note that in the optimization function the training data only show up in inner produts

Support vector machines

Linearly separable classes (VI)

Classifying new data points

- after finding the optimal λ, we can classify new data points, evaluating the sign of

$$y(\mathbf{x}) = \mathbf{x}^T \mathbf{w} + b = \sum_n \lambda_n y_n \mathbf{x}^T \mathbf{x}_n + b \tag{1}$$

  - Any training point with $\lambda_n = 0$ plays no role in making predictions
    - linear regression estimates a set of parameters and then the training points are not used for making predictions
    - k-Nearest Neighbour uses ALL training points to make predictions
    - SVMs is something inbetween: uses only points with $\lambda_n \neq 0$ (support vectors) to make predictions: sparse kernel machine

Support vector machines

Linearly separable classes (VII)

Classifying new data points

- note that $\mathbf{w}$ does not need to be computed explicitly... but we need b
- note that any support vector satisfies (from the KKT conditions or from the very own formulation of SVMs) $y_m y(\mathbf{x}_m) = 1 \Leftrightarrow y_m(\mathbf{w}^t \mathbf{x}_m + b) = 1$. Using (1) we get $y_m(\sum_n \lambda_n y_n \mathbf{x}_m^T \mathbf{x}_n + b) = 1$, where $\mathbf{x}_m$ is any support vector
- $b = y_m - \sum_n \lambda_n y_n \mathbf{x}_m^T \mathbf{x}_n$
- it is numericaly more stable to average over ALL support vectors: $b = \frac{1}{N_S} \sum_{m \in \mathcal{S}} \left\{ y_m - \sum_{n \in \mathcal{S}} \lambda_n y_n \mathbf{x}_m^T \mathbf{x}_n \right\}$, with m and n running only over support vectors

Support vector machines

Linearly separable classes (VIII)

A geometric interpretation

- find the convex hull of both sets of points
- find the two closest points in the two convex hulls (the convex hull is the smallest convex set containing the points)
- construct the plane that bisects these two points

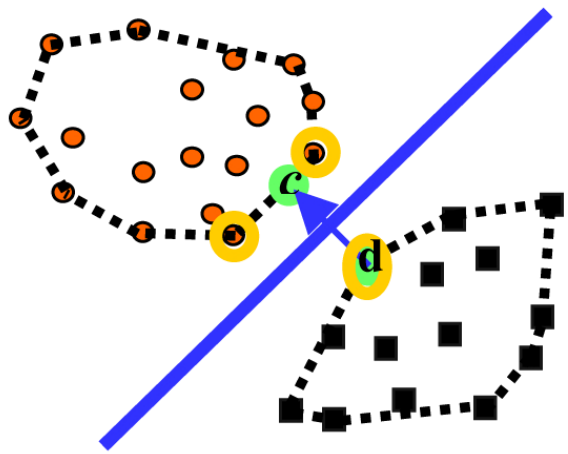


Figure: from [3]

Support vector machines

Linearly separable classes (IX)

A geometric interpretation

- The closest points in the convex hulls can be found by solving the following quadratic problem

$$
\begin{aligned}
\text{argmin}_{\alpha} \quad & \frac{1}{2}||c-d||^2 \\
s.t \quad & c = \sum_{\mathbf{x}_i \in \mathcal{C}_1} \alpha_i \mathbf{x}_i \\
& d = \sum_{\mathbf{x}_i \in \mathcal{C}_2} \alpha_i \mathbf{x}_i \\
& \sum_{\mathbf{x}_i \in \mathcal{C}_1} \alpha_i = 1 \\
& \sum_{\mathbf{x}_i \in \mathcal{C}_2} \alpha_i = 1 \\
& \alpha_i \geq 0, i = 1, \cdots, N
\end{aligned}
$$

- it can be shown to be equivalent to the maximum margin approach

Support vector machines

Non-linearly separable classes (I)

When in possession of a nearly separable dataset, a simple linear separator is bound to misclassify some points. But the question that we have to ask ourselves is if the non-linearly separable data portraits some intrinsic property of the problem (in which case a more complex classifier, allowing more general boundaries between classes may be more appropriate) or if it can be interpreted as the result of noisy points (measurement errors, uncertainty in class membership, etc), in which case keeping the linear separator and accept some errors is more natural.

- we will keep the linear boundary for now and accept errors in the training data

Support vector machines

Non-linearly separable classes (II)

- allow some of the training points to be misclassified
- penalize the number of misclassifications
- introduce slack variables $\xi_n \geq 0, n = 1, \ldots, N$

$$\xi_n = 0 \quad \mathbf{x}_n \text{ on the correct side of the margin}$$

$$\xi_n = |y_n - y(\mathbf{x}_n)| \quad \text{otherwise}$$

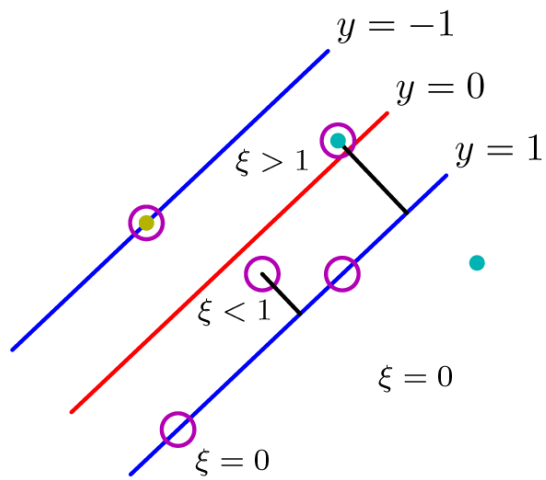


Figure: from [1]

Support vector machines

Non-linearly separable classes (III)

- replace exact constraints with $y_n y(\mathbf{x}_n) \geq 1 - \xi_n \Leftrightarrow y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n$
- the goal could now be to minimize $\frac{1}{2}||\mathbf{w}||^2 + C \sum_{n=1}^{N} sign(\xi_n)$
- The constraint parameter C controls the tradeoff between the dual objectives of maximizing the margin of separation and minimizing the misclassification error
- For an error to occur, the corresponding ξ_n must exceed unity, so $\sum_{n=1}^{N} sign(\xi_n)$ is an upper bound on the number of the training errors
- optimization of the above is difficult since it involves a discontinuous function $sign()$
- to optimize a closely related cost function $\frac{1}{2}||\mathbf{w}||^2 + C \sum_{n=1}^{N} \xi_n$

Support vector machines

Non-linearly separable classes (III)

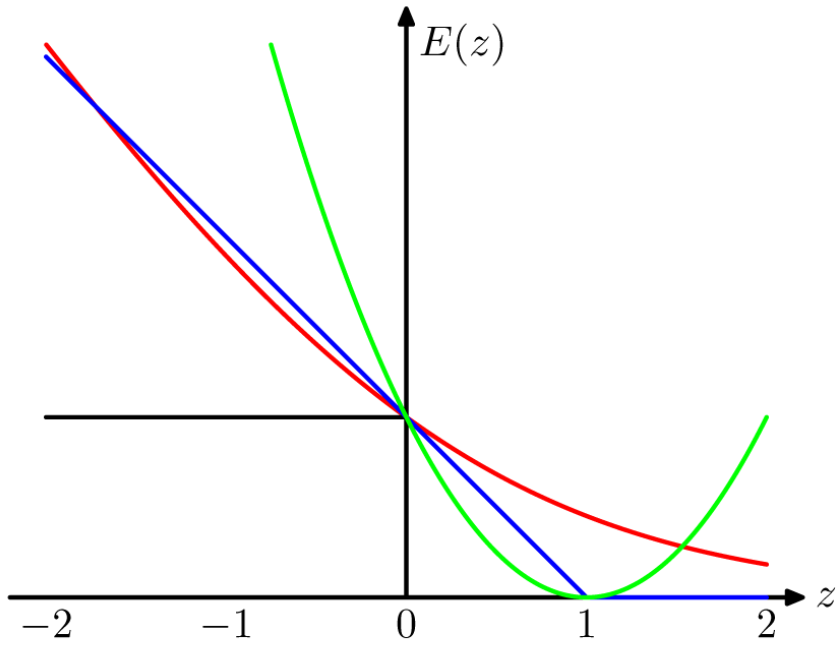


Figure: from [1]

Support vector machines

Non-linearly separable classes (IV)

- $L(\mathbf{w}, b, \xi, \mu, \lambda) =$
  $\frac{1}{2}||\mathbf{w}||^2 + C\sum \xi_n - \sum \lambda_n(y_n y(\mathbf{x}_n) - 1 + \xi_n) - \sum \mu_n \xi_n$
- KKT conditions:

$$\begin{aligned}
\lambda_n &\geq 0 \\
y_n y(\mathbf{x}_n) - 1 + \xi_n &\geq 0 \\
\lambda(y_n y(\mathbf{x}_n) - 1 + \xi_n) &= 0 \\
\mu_n &\geq 0 \\
\xi_n &\geq 0 \\
\mu_n \xi_n &= 0
\end{aligned}$$

Support vector machines

Non-linearly separable classes (Va)

- $\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_n \lambda_n y_n \mathbf{x}_n$
- $\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_n \lambda_n y_n = 0$
- $\frac{\partial L}{\partial \xi_n} = 0 \Rightarrow \lambda_n = C - \mu_n$
- Dual Lagrangian form:
  $\text{argmax}_\lambda L(\lambda) = \sum \lambda_{n=1}^N - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \lambda_n \lambda_m y_n y_m \mathbf{x}_m^T \mathbf{x}_n$
  subject to:
  $0 \leq \lambda_n \leq C$
  $\sum_{n=1}^N \lambda_n y_n = 0$
- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound C on λ_n now
- a QP solver can be used to find λ_n

Support vector machines

Non-linearly separable classes (Vb)

The SMO algorithm

- Consider solving the unconstrained opt problem:

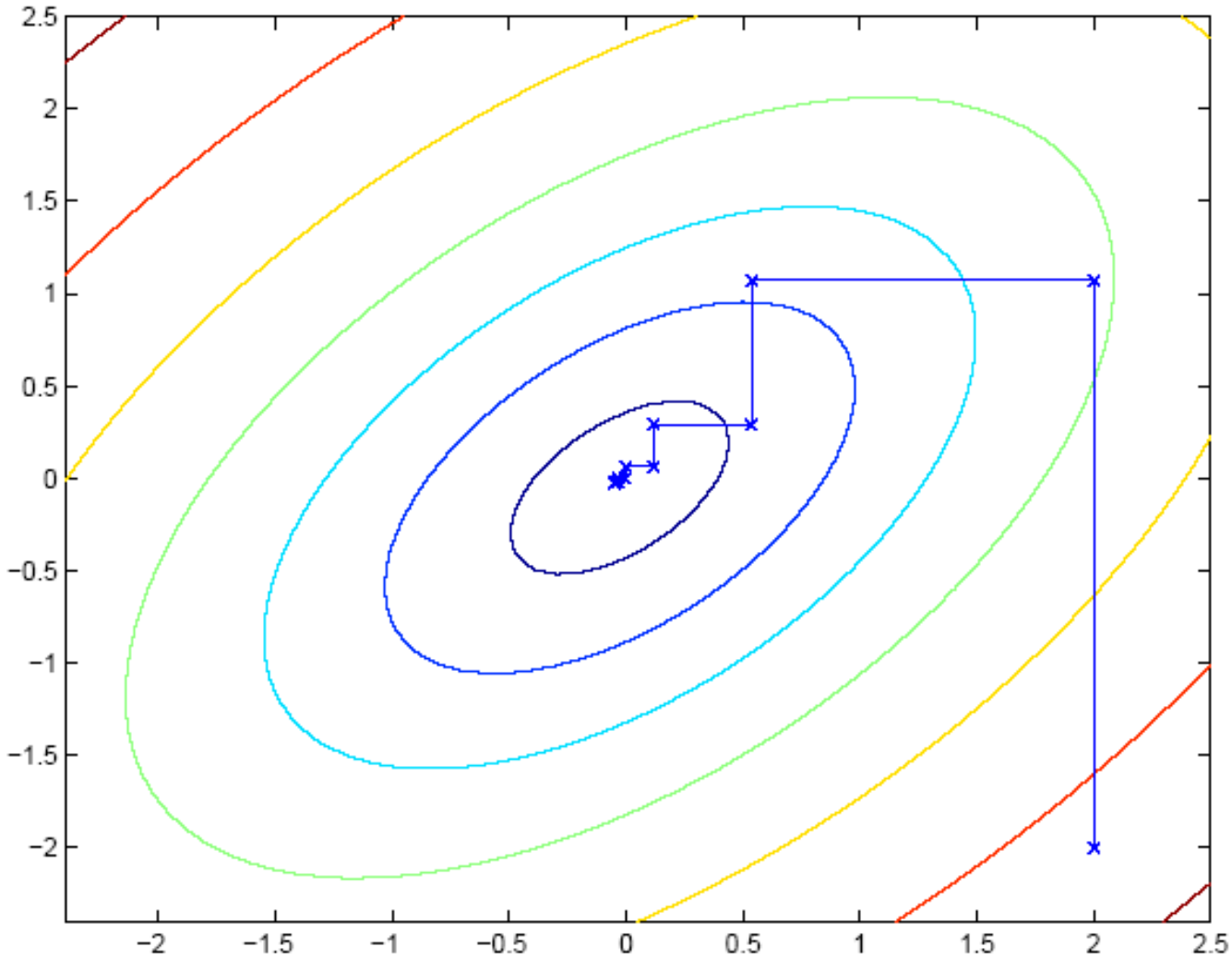
$$\max_{\boldsymbol{\alpha}} W(\alpha_1, \cdots, \alpha_m)$$

- EM (to be discussed)
- gradient ascend
- Coordinate ascend

Support vector machines

Non-linearly separable classes (Vc)

Coordinate ascend



Support vector machines

Non-linearly separable classes (Vc)

Sequential minimal optimization

- Constrained optimization:
  $\text{argmax}_\lambda L(\lambda) = \sum \lambda_{n=1}^{N} - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m y_n y_m \mathbf{x}_m^T \mathbf{x}_n$
  subject to:
  $0 \leq \lambda_n \leq C$
  $\sum_{n=1}^{N} \lambda_n y_n = 0$
- Question: can we do coordinate along one direction at a time (i.e., hold all $\lambda_{[-i]}$ fixed, and update λ_i?)

Support vector machines

Non-linearly separable classes (Vd)

The SMO algorithm

- Repeat till convergence
  - Select some pair λ_i and λ_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).
  - Re-optimize $L(\lambda)$ with respect to λ_i and λ_j, while holding all the other λ_k's $(k \neq i; j)$ fixed.

Support vector machines

Non-linearly separable classes (VI)

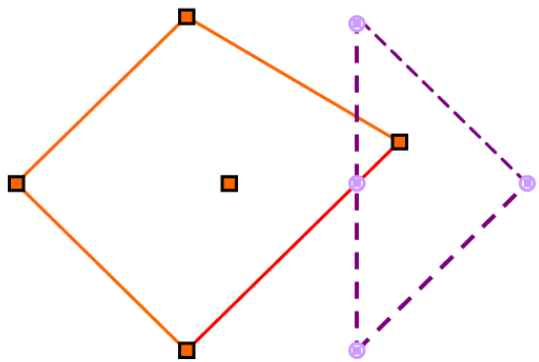
- Predictions: $y(\mathbf{x}) = \sum_{n=1}^{N} \lambda_n y_n \mathbf{x}^T \mathbf{x}_n + b$
- $b = \frac{1}{N_{\mathcal{M}}} \sum_{n \in \mathcal{M}} (y_n - \sum_{m \in \mathcal{S}} \lambda_m y_m \mathbf{x}_m^t \mathbf{x}_n)$
- where $\mathcal{M}$ denotes the set of indices of data having $0 < \lambda_n < C$
- and $\mathcal{S}$ denotes the set of indices of data having $\lambda_n \neq 0$ (set of support vectors)

Support vector machines

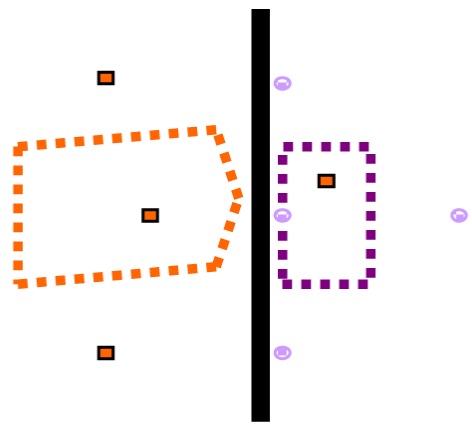
Non-linearly separable classes (VII)

A geometric interpretation

- find the reduced convex hull of both sets of points
- find the two closest points in the two reduced convex hulls (the convex hull is the smallest convex set containing the points)
- construct the plane that bisects these two points



(a) An example of two non-linearly separable datasets (from [3]).



(b) Reduced Convex Hull approach (from [3])

Support vector machines

Non-linearly separable classes (VIII)

A geometric interpretation

- The closest points in the reduced convex hulls can be found by solving the following quadratic problem

$$
\begin{aligned}
\text{argmin}_{\alpha} \quad & \frac{1}{2}||c-d||^2 \\
s.t \quad & c = \sum_{\mathbf{x}_i \in \mathcal{C}_1} \alpha_i \mathbf{x}_i \\
& d = \sum_{\mathbf{x}_i \in \mathcal{C}_2} \alpha_i \mathbf{x}_i \\
& \sum_{\mathbf{x}_i \in \mathcal{C}_1} \alpha_i = 1 \\
& \sum_{\mathbf{x}_i \in \mathcal{C}_2} \alpha_i = 1 \\
& 0 \leq \alpha_i \leq D, i = 1, \cdots, N
\end{aligned}
$$

- it can be shown to be equivalent to the maximum margin approach

Support vector machines

Non-linearly separable classes (IX)

- show that for a linearly separable dataset, the solution for the formulation with ξ's may be different from the solution for the formulation without the ξ's.
- what is the solution for the formulation without the ξ's for a non-linearly separable dataset?

Support vector machines

Non-linearly separable classes (X)

- Dual Lagrangian form:
  $L(\lambda) = \sum \lambda_{n=1}^{N} - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m y_n y_m \mathbf{x}_m^T \mathbf{x}_n$
  subject to:
  $0 \leq \lambda_n \leq C$
  $\sum_{n=1}^{N} \lambda_n y_n = 0$

- $b = \frac{1}{N_{\mathcal{M}}} \sum_{m \in \mathcal{M}} (y_m - \sum_{s \in \mathcal{S}} \lambda_s y_s \mathbf{x}_s^T \mathbf{x}_m)$
- Predictions: $y(\mathbf{x}) = \sum_{n=1}^{N} \lambda_n y_n \mathbf{x}^T \mathbf{x}_n + b$

- Notes:
  - N unknowns: λ's
  - the input vector $\mathbf{x}$ enters only in the form of scalar products
  - the dimension of the data has almost no impact on the complexity

Support vector machines

The design of nonlinear classifiers

- The design of a nonlinear classifier for the original data space $\mathbf{x} \in \mathbf{R}^d$ can be accomplished with the design of a linear classifier in a suitable space $\hat{\mathbf{x}} \in \mathbf{R}^D$
  - usually $D > d$ but that is not always necessary
- Examples
  1. the search for a boundary $x_1^2 + x_2^2 = const$ in $\mathbf{R}^2$ can be found as a linear boundary in the space defined by the variables $r = \sqrt{x_1^2 + x_2^2}$ and $\theta = arctan(y/x)$. In fact it could be simplified to the search in $\mathbf{R}$ defined just by the variable r
  2. the search for a boundary $w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2 + w_6 = 0$ in $\mathbf{R}^2$ can be found as a linear boundary in the space defined by the variables $\hat{x_1} = x_1$, $\hat{x_2} = x_2$, $\hat{x_3} = x_1 x_2$, $\hat{x_4} = x_1^2$, $\hat{x_5} = x_2^2$

Support vector machines

The design of nonlinear classifiers

- to design a nonlinear SVM we just define a new space $\Phi(\mathbf{x})$ where our problem becomes linear
- apply the linear SVM in the new space $\Phi(\mathbf{x})$ (the linearly separable version or the nonlinearly separable version)
- the changes in our formulations are minimum:
  - just replace $\mathbf{x}_i^T\mathbf{x}_j$ by $\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$
- Define $k(\mathbf{x}_i,\mathbf{x}_j)=\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$. The nonlinear SVMs become:
  $L(\lambda)=\sum\lambda_{n=1}^N-\frac{1}{2}\sum_{n=1}^N\sum_{m=1}^N\lambda_n\lambda_m y_n y_m\ k(\mathbf{x}_m,\mathbf{x}_n)$
  subject to:
  $0\leq\lambda_n\leq C$
  $\sum_{n=1}^N\lambda_n y_n=0$
- $b=\frac{1}{N_{\mathcal{M}}}\sum_{m\in\mathcal{M}}(y_m-\sum_{s\in\mathcal{S}}\lambda_s y_s\ k(\mathbf{x}_s,\mathbf{x}_m))$
- Predictions: $y(\mathbf{x})=\sum_{n=1}^N\lambda_n y_n\ k(\mathbf{x},\mathbf{x}_n)+b$
- The new space $\Phi(\mathbf{x})$ should capture the nonlinearity intrinsic to the data (and not the noise). In the simplest case $\Phi(\mathbf{x})=\mathbf{x}$ and $k(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$

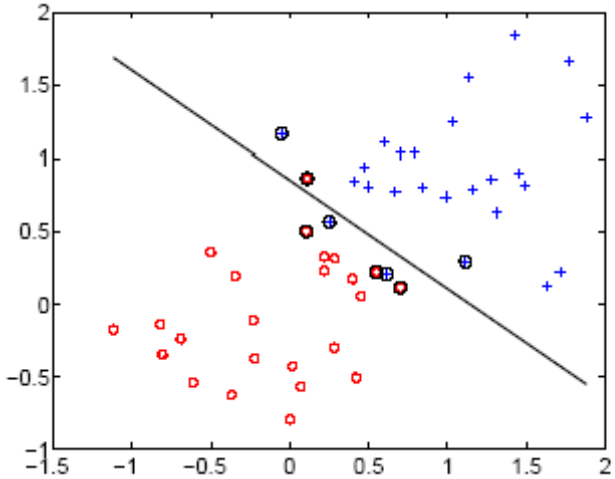
Kernel methods

The design of nonlinear classifiers

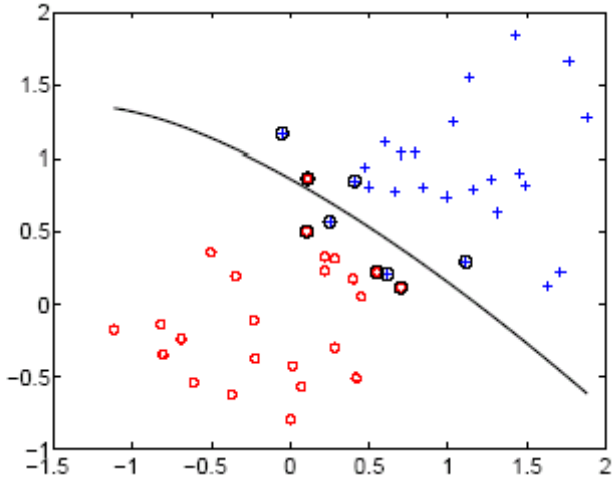
- It would be nice to be able to compute $k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$ without needing to use the transformed space $\Phi(\mathbf{x})$
- Mercer's theorem
  Let $\mathbf{x} \in \mathbf{R}^d$ and a mapping $\Phi(\mathbf{x})$
  $\mathbf{x} \to \Phi(\mathbf{x}) \in H$, where H is a Euclidean space (linear space equipped with inner product). Then
  $\Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_2) = k(\mathbf{x}_1, \mathbf{x}_2)$, where $k(\mathbf{x}_1, \mathbf{x}_2)$ is a symmetric function satisfying $\int k(\mathbf{x}_1, \mathbf{x}_2) g(\mathbf{x}_1) g(\mathbf{x}_2) d\mathbf{x}_1 d\mathbf{x}_2 \geq 0$, for any $g(\mathbf{x}), \mathbf{x} \in \mathbf{R}^d$ such that $\int g^2(\mathbf{x}) d\mathbf{x} < \infty$
- Note that Mercer's theorem does not tell us how to find this space.
- A necessary and sufficient condition for a function $k(\mathbf{x}, \mathbf{x})$ to be a valid kernel is that the Gram Matrix K, whose elements are given by $k(\mathbf{x}_n, \mathbf{x}_m)$, should be a positive semidefinite matrix for all possible choices of the set $\{\mathbf{x}_n\}$

Kernel methods

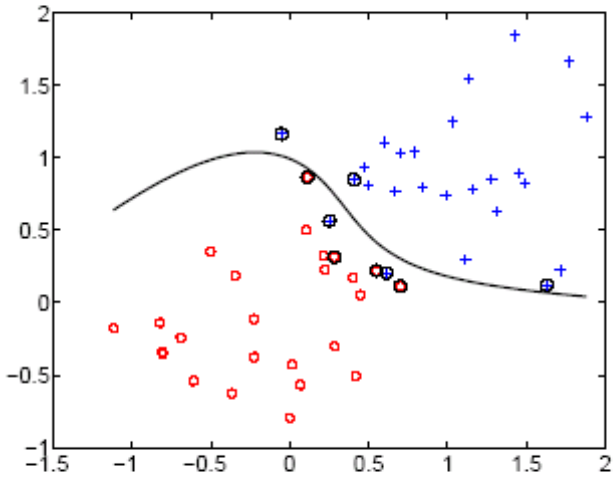
SVM examples



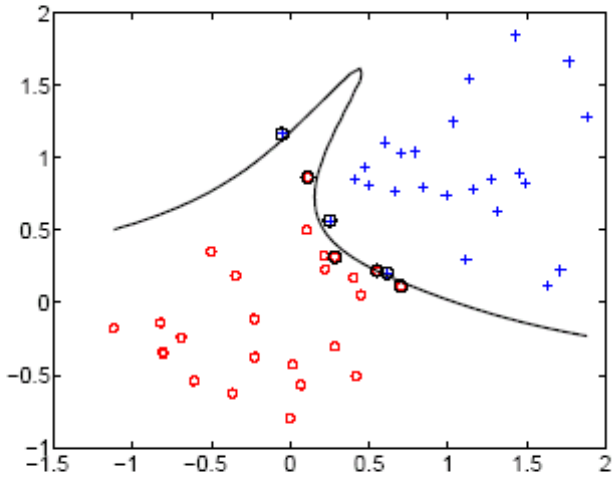
linear



2^{nd} order polynomial



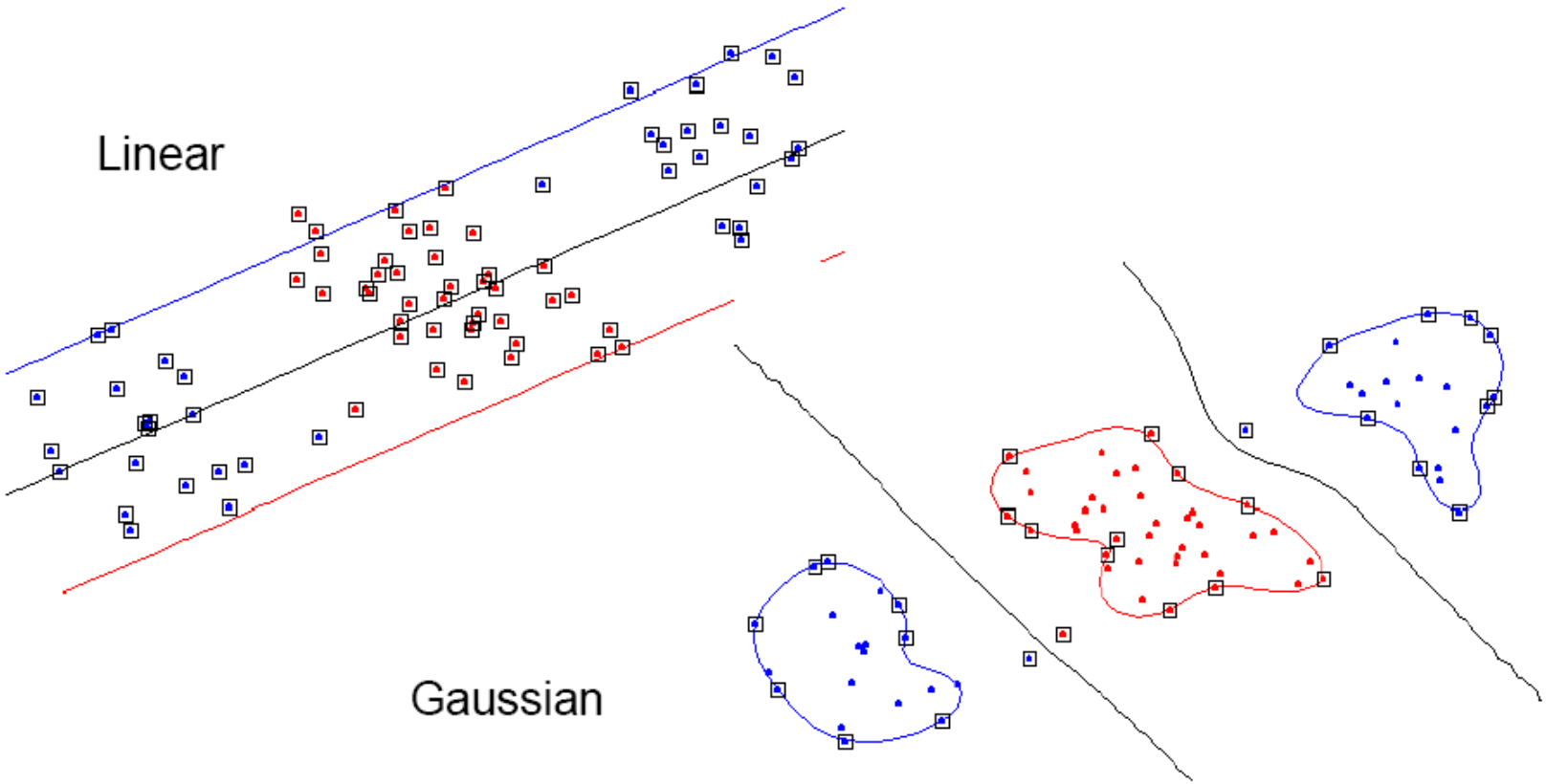
4^{th} order polynomial



8^{th} order polynomial

Kernel methods

SVM examples



Kernel methods

The design of nonlinear classifiers

- many models can be reformulated in terms of a dual representation in which the kernel function arises naturally
- linear regression
- kernel PCA
- nearest neighbour
- etc

Beyond basic Linear Regression

Beyond basic LR

Dual Variables in Regression

- $(\mathbf{X}^t\mathbf{X} + \lambda I)\mathbf{w} = \mathbf{X}^t\mathbf{y}$
- $\mathbf{w} = \lambda^{-1}\mathbf{X}^t(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{X}^t\boldsymbol{\alpha}$
- showing that $\mathbf{w}$ can be written as a linear combination of the training points, $\mathbf{w} = \sum_{n=1}^{N} = \alpha_n\mathbf{x}_n$, with $\boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$
- The elements of $\boldsymbol{\alpha}$ are called the dual variables.
- $$\begin{aligned} \boldsymbol{\alpha} &= \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &\Rightarrow \lambda\boldsymbol{\alpha} = \mathbf{y} - \mathbf{X}\mathbf{X}^t\boldsymbol{\alpha} \\ &\Rightarrow (\mathbf{X}\mathbf{X}^t + \lambda I)\boldsymbol{\alpha} = \mathbf{y} \\ &\Rightarrow \boldsymbol{\alpha} = (\mathbf{G} + \lambda I)^{-1}\mathbf{y} \end{aligned}$$
- $\mathbf{G} = \mathbf{X}\mathbf{X}^t$ or, component-wise, $G_{ij} =< \mathbf{x}_i, \mathbf{x}_j >$.
- The resulting prediction function is given by $g(\mathbf{x}) =< \mathbf{w}, \mathbf{x} >= \left\langle \sum_{n=1}^{N} \alpha_n\mathbf{x}_n, \mathbf{x} \right\rangle = \sum_n \alpha_n < \mathbf{x}_n, \mathbf{x} >$

Kernel methods

An example of a kernel algorithm

Idea: classify points $\mathbf{x}$ in the feature space according to which of the two class means is closer.

$c_+ = \frac{1}{N_+} \sum_{i:y_i=+1} \mathbf{x}_i \quad c_- = \frac{1}{N_-} \sum_{i:y_i=-1} \mathbf{x}_i$

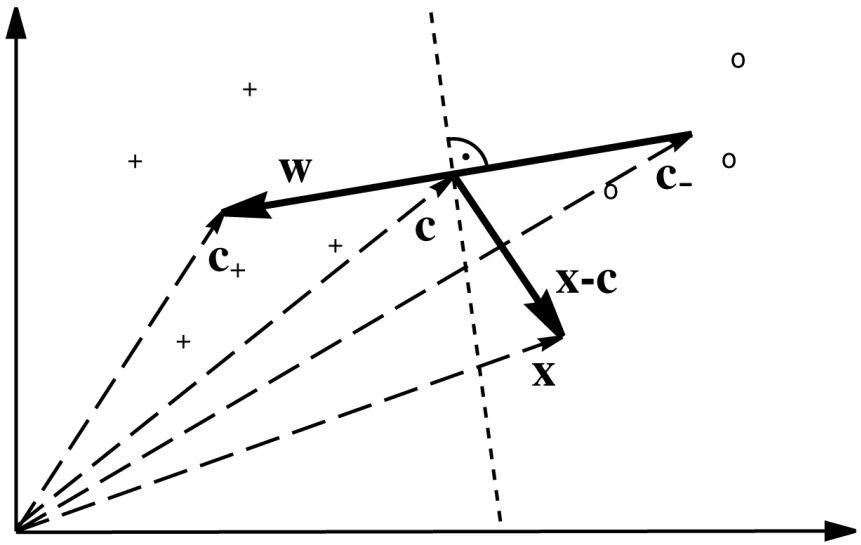


Figure: from [?]

Compute the sign of the dot product between $\mathbf{w} = \mathbf{c}_+ - \mathbf{c}_-$ and $\mathbf{x} - \mathbf{c}$.

Kernel methods

An example of a kernel algorithm

$$
\begin{aligned}
f(\mathbf{x}) &= sgn\left(\frac{1}{N_+}\sum_{i:y_i=+1} < \mathbf{x}, \mathbf{x}_i > -\frac{1}{N_-}\sum_{i:y_i=-1} < \mathbf{x}, \mathbf{x}_i > +b\right) \\
&= sgn\left(\frac{1}{N_+}\sum_{i:y_i=+1} k(\mathbf{x}, \mathbf{x}_i) - \frac{1}{N_-}\sum_{i:y_i=-1} k(\mathbf{x}, \mathbf{x}_i) + b\right)
\end{aligned}
$$

where $b =$
$\frac{1}{2}\left(\frac{1}{N_-^2}\sum_{(i,j):y_i=y_j=-1} k(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{N_+^2}\sum_{(i,j):y_i=y_j=+1} k(\mathbf{x}_i, \mathbf{x}_j)\right)$

Kernel methods

Typical kernels

- any algorithm that only depends on dot products can benefit form the kernel trick
- think of the kernel as a nonlinear similarity measure
- examples of common kernels:
  - Linear $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{x}_2$
  - Polynomial $k(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1^T \mathbf{x}_2 + c)^d$
  - Gaussian $k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\gamma ||\mathbf{x}_1 - \mathbf{x}_2||^2)$
- kernels are also known as covariance functions

Kernel methods

Constructing kernels

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

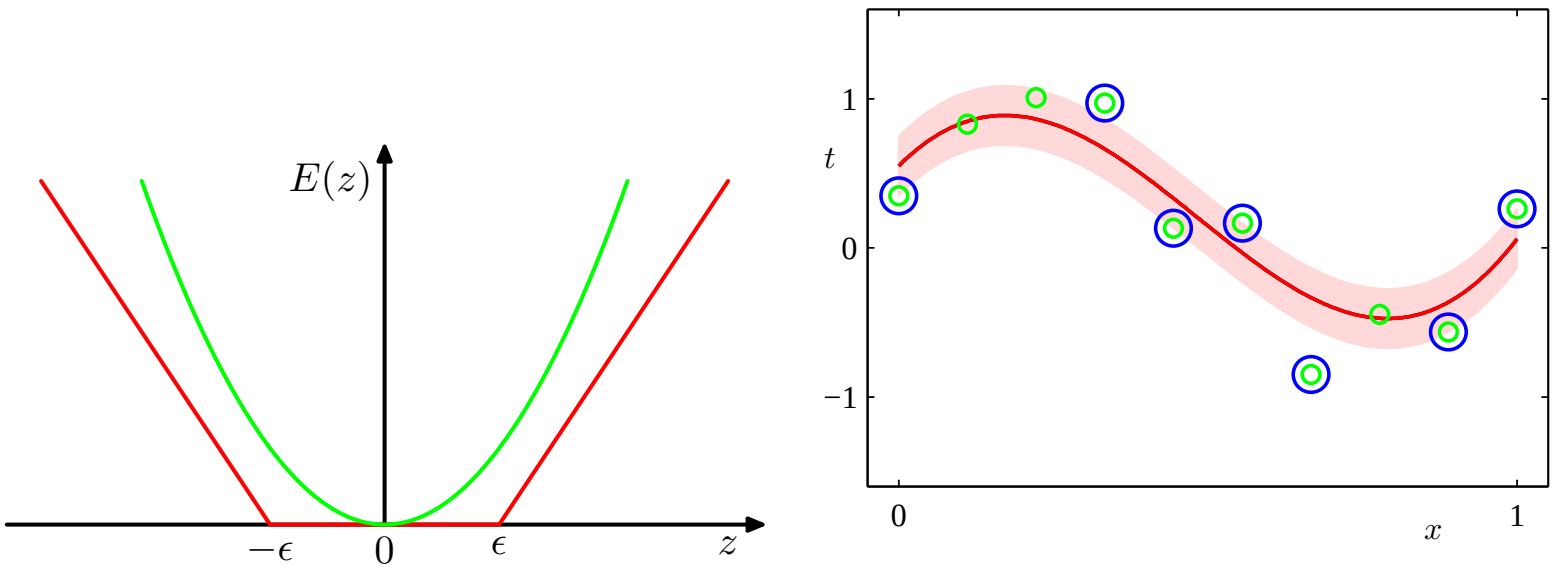
- $k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}')$, $c \in \mathbf{R}_+$
- $k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')$, $f(.)$ is any function
- $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$
- $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$
- $k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}'))$

Regression Support Vector Machine

Regression Support Vector Machine

- $\min_{\mathbf{w},b} C \sum_n E_\epsilon(y_n - f(\mathbf{x}_n)) + \frac{\lambda}{2}||\mathbf{w}||^2$

$$E_\epsilon(z) = \begin{cases} 0 \text{ if } |z| < \epsilon \\ |z| - \epsilon, \text{ otherwise} \end{cases}$$



(a) $\epsilon-$insensitive error function. (b) RVM result.

Figure: from [1]

Multiclass Support Vector Machine

Reduction techniques

Conventional approaches

- One-against-All
  K two-class problems
- Pairwise
  $K(K-1)/2$ two-class problems
- Decision-Tree-Based
- DAG (Directed Acyclic Graph)
- Error-Correcting Output Codes

Multiclass Support Vector Machine

Reduction techniques

Conventional approaches

- apply binary classifier 1 to test example and get prediction $F1$ $(0/1)$
- apply binary classifier 2 to test example and get prediction $F2$ $(0/1)$
- ...
- apply binary classifier M to test example and get prediction FM $(0/1)$
- use all M classifications to get the final multiclass classification $1..K$

Multiclass Support Vector Machine

Reduction techniques

A lesser-known approach for solving multiclass problems via a **Single binary classifier reduction**

- obtained by embedding the original problem in a higher-dimensional space consisting of the original features, as well as one or more other dimensions determined by fixed vectors, termed here extension features.
- This embedding is implemented by replicating the training set points so that a copy of the original point is concatenated with each of the extension features' vectors.
- The binary labels of the replicated points are set to maintain a particular structure in the extended space.
- This construction results in an instance of an artificial binary problem, which is fed to a binary learning algorithm that outputs a single soft binary classifier.
- To classify a new point, the point is replicated and extended similarly and the resulting replicas are fed to the soft binary classifier, which generates a number of signals, one for each replica. The class is determined as a function of these signals.

Multiclass Support Vector Machine

All-at-Once Support Vector Machines

For a K-class problem we define the decision function for class i by

- $\mathbf{w}_i^T \mathbf{x} + b_i > \mathbf{w}_k^T \mathbf{x} + b_k$, for $k \neq i, k = 1, \ldots, K$
- The L1 soft-margin support vector machine can be obtained by minimizing

$$\min_{\mathbf{w},\mathbf{b},\xi} = \frac{1}{2} \sum_{k=1}^{K} ||\mathbf{w}_k||^2 + C \sum_{n=1}^{N} \sum_{k=1, k \neq y_n}^{K} \xi_{n,k}$$

- subject to

$$(\mathbf{w}_{y_n} - \mathbf{w}_k)^T \mathbf{x}_k + b_{y_n} - b_k \geq 1 - \xi_{n,k}$$

for $k \neq y_n, k = 1, \ldots, K, n = 1, \ldots, N$

References


Christopher M. Bishop
Pattern recognition and machine learning,
Springer, 2006 .


Sergios Theodoridis and Konstantinos Koutroumbas
Pattern recognition
Elsevier, Academic Press, 2003.


Kristin P. Bennet and Campbell
Support Vector Machines: Hype or Hallelujah?
SIGKDD Explorations, vol. 2, 2000


Michael E. Mavroforakis and Sergios Theodoridis
A Geometric Approach to Support Vector Machine (SVM)
Classification
IEEE Transactions on Neural Networks, 2006