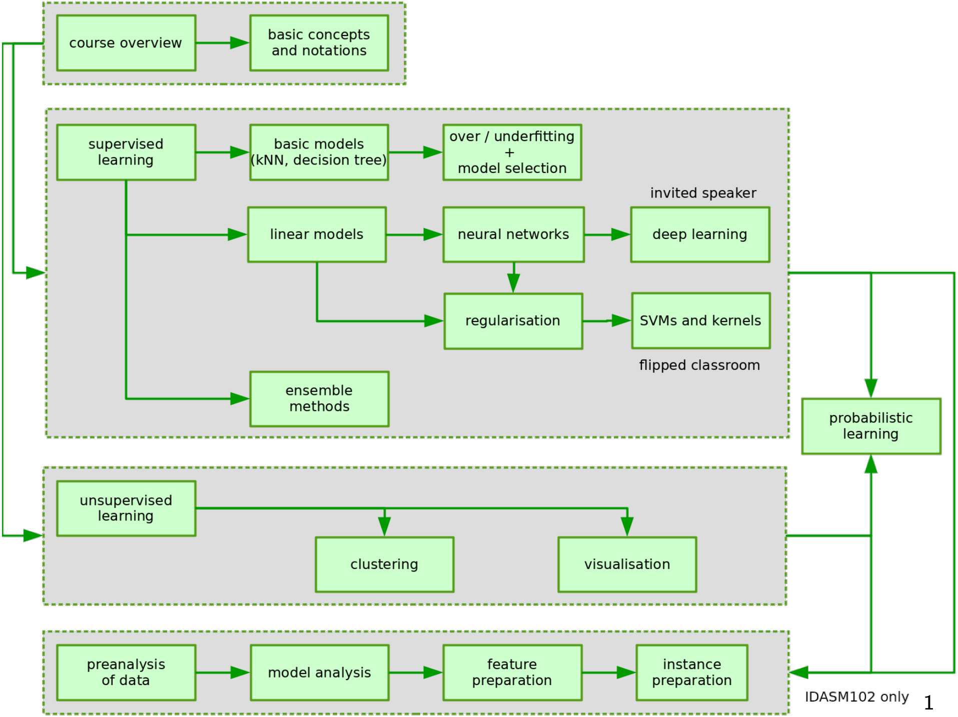
Machine Learning: Lesson 10

Support Vector Machines and Kernels

Benoît Frénay - Faculty of Computer Science

UNIVERSITÉ DE NAMUR


course overview
basic concepts and notations
supervised learning
basic models (kNN, decision tree)
over / underfitting + model selection
invited speaker
linear models
neural networks
deep learning
regularisation
SVMs and kernels
flipped classroom
ensemble methods
probabilistic learning
unsupervised learning
clustering
visualisation
preanalysis of data
model analysis
feature preparation
instance preparation
IDASM102 only

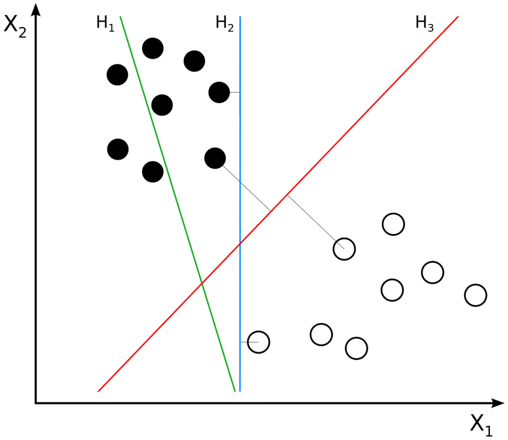
Outline of this Lesson

- maximum margin classifiers
- linear SVMs for separable data
- linear SVMs for non-separable data
- kernels and similarity measures
- kernelised SVMs for non-linear data
- going further with kernels

Maximum Margin Classifiers

Comparing Binary Classifiers in Terms of Margin

$$\text{linear classifier} = \begin{cases} y_i = +1 & \text{if } \mathbf{w} \cdot \mathbf{x}_i + b = \sum_{j=1}^{d} w_i x_{ij} + b > 0 \\ y_i = -1 & \text{if } \mathbf{w} \cdot \mathbf{x}_i + b = \sum_{j=1}^{d} w_i x_{ij} + b < 0 \end{cases}$$



http://en.wikipedia.org/wiki/Support_vector_machine

Definition of the Margin for Binary Classifiers

margin = (width of the) largest region that separates both classes perfectly

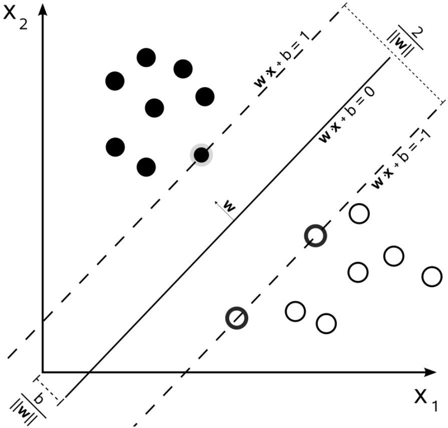
black class on $\mathbf{w} \cdot \mathbf{x} + b = 1$

$$\mathbf{w} \cdot \left(\mathbf{x}_0 + \frac{\delta}{2} \hat{\mathbf{w}} \right) + b = 1$$

white class on $\mathbf{w} \cdot \mathbf{x} + b = -1$

$$\mathbf{w} \cdot \left(\mathbf{x}_0 - \frac{\delta}{2} \hat{\mathbf{w}} \right) + b = -1$$

margin is given by $\delta = \frac{2}{\|\mathbf{w}\|}$



http://en.wikipedia.org/wiki/Support_vector_machine

Definition of the Margin for Binary Classifiers

margin = (width of the) largest region that separates both classes perfectly

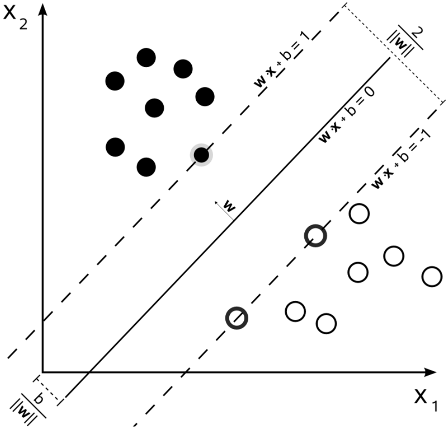
black class on $\mathbf{w} \cdot \mathbf{x} + b = 1$

$$\mathbf{w} \cdot \left(\mathbf{x}_0 + \frac{\delta}{2} \hat{\mathbf{w}} \right) + b = 1$$

white class on $\mathbf{w} \cdot \mathbf{x} + b = -1$

$$\mathbf{w} \cdot \left(\mathbf{x}_0 - \frac{\delta}{2} \hat{\mathbf{w}} \right) + b = -1$$

margin is given by $\delta = \frac{2}{\|\mathbf{w}\|}$



http://en.wikipedia.org/wiki/Support_vector_machine

Definition of the Margin for Binary Classifiers

margin = (width of the) largest region that separates both classes perfectly

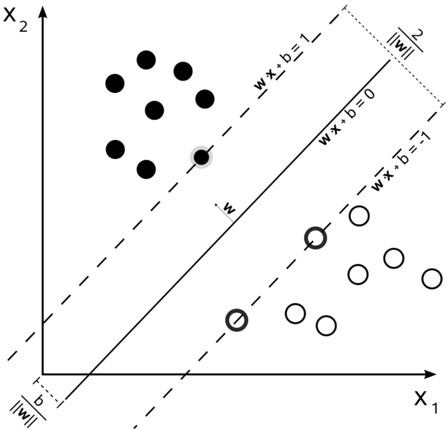
black class on $\mathbf{w} \cdot \mathbf{x} + b = 1$

$$\mathbf{w} \cdot \left(\mathbf{x}_0 + \frac{\delta}{2} \hat{\mathbf{w}} \right) + b = 1$$

white class on $\mathbf{w} \cdot \mathbf{x} + b = -1$

$$\mathbf{w} \cdot \left(\mathbf{x}_0 - \frac{\delta}{2} \hat{\mathbf{w}} \right) + b = -1$$

margin is given by $\delta = \frac{2}{\|\mathbf{w}\|}$



http://en.wikipedia.org/wiki/Support_vector_machine

Definition of the Margin for Binary Classifiers

margin = (width of the) largest region that separates both classes perfectly

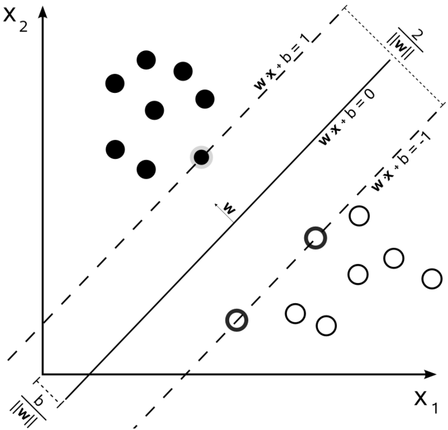
black class on $\mathbf{w} \cdot \mathbf{x} + b = 1$

$$\mathbf{w} \cdot \left(\mathbf{x}_0 + \frac{\delta}{2} \hat{\mathbf{w}} \right) + b = 1$$

white class on $\mathbf{w} \cdot \mathbf{x} + b = -1$

$$\mathbf{w} \cdot \left(\mathbf{x}_0 - \frac{\delta}{2} \hat{\mathbf{w}} \right) + b = -1$$

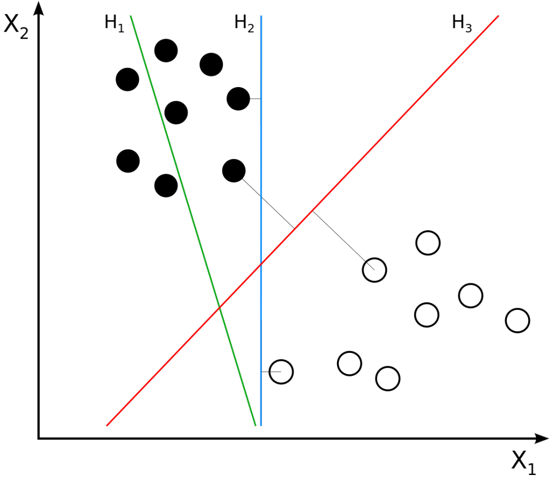
margin is given by $\delta = \frac{2}{\|\mathbf{w}\|}$



http://en.wikipedia.org/wiki/Support_vector_machine

The Maximum Margin Principle for Separable Data

best linear classifier = separates data perfectly with largest margin $\delta = \frac{2}{\|\mathbf{w}\|}$



http://en.wikipedia.org/wiki/Support_vector_machine

Linear SVMs for Separable Data

Derivation of Linear Support Vector Machines

Maximum margin linear classifier

for linearly separable data, maximising the margin is equivalent to

$$\begin{array}{lll} \max & \delta = \frac{2}{\|\mathbf{w}\|} & \\ \text{s.t.} & \mathbf{w} \cdot \mathbf{x}_i + b \geq +1 & \text{for each instance s.t. } t_i = +1 \\ & \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 & \text{for each instance s.t. } t_i = -1 \end{array}$$

Primal form of linear SVMs for separable data

for linearly separable data, maximising the margin is equivalent to

$$\begin{array}{ll} \min & \|\mathbf{w}\| \\ \text{s.t.} & t_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) \geq 1 \end{array}$$

Derivation of Linear Support Vector Machines

Maximum margin linear classifier

for linearly separable data, maximising the margin is equivalent to

$$
\begin{array}{lll}
\max & \delta = \frac{2}{\|\mathbf{w}\|} & \\
\text{s.t.} & \mathbf{w} \cdot \mathbf{x}_i + b \geq +1 & \text{for each instance s.t. } t_i = +1 \\
 & \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 & \text{for each instance s.t. } t_i = -1
\end{array}
$$

Primal form of linear SVMs for separable data

for linearly separable data, maximising the margin is equivalent to

$$
\begin{array}{ll}
\min & \|\mathbf{w}\| \\
\text{s.t.} & t_i\left(\mathbf{w} \cdot \mathbf{x}_i + b\right) \geq 1
\end{array}
$$

Derivation of Linear Support Vector Machines

Dual variables

primal form of linear SVMs can be rewritten with dual variables $\alpha_i \geq 0$ s.t.

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i t_i \mathbf{x}_i \quad \text{where} \quad \sum_{i=1}^{n} \alpha_i t_i = 0$$

α_i = "weight of instance $\mathbf{x}_i$ in expression of $\mathbf{w}$" ($\times \pm 1$ if class $t_i = \pm 1$)

Primal variables vs. dual variables

in primal form: prediction $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{j=1}^{d} w_j x_j + b$

- the contribution of each feature is weighted by w_j

in dual form: prediction $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i (\mathbf{x}_i \cdot \mathbf{x}) + b$

- the contribution of each instance is weighted by α_i

Derivation of Linear Support Vector Machines

Dual variables

primal form of linear SVMs can be rewritten with dual variables $\alpha_i \geq 0$ s.t.

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i t_i \mathbf{x}_i \quad \text{where} \quad \sum_{i=1}^{n} \alpha_i t_i = 0$$

α_i = "weight of instance $\mathbf{x}_i$ in expression of $\mathbf{w}$" ($\times \pm 1$ if class $t_i = \pm 1$)

Primal variables vs. dual variables

in primal form: prediction $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{j=1}^{d} w_j x_j + b$

- the contribution of each **feature** is weighted by w_j

in dual form: prediction $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i (\mathbf{x}_i \cdot \mathbf{x}) + b$

- the contribution of each **instance** is weighted by α_i

Derivation of Linear Support Vector Machines

Dual variables

primal form of linear SVMs can be rewritten with dual variables $\alpha_i \geq 0$ s.t.

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i t_i \mathbf{x}_i \quad \text{where} \quad \sum_{i=1}^{n} \alpha_i t_i = 0$$

α_i = "weight of instance $\mathbf{x}_i$ in expression of $\mathbf{w}$" ($\times \pm 1$ if class $t_i = \pm 1$)

Primal variables vs. dual variables

in primal form: prediction $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{j=1}^{d} w_j x_j + b$

- the contribution of each **feature** is weighted by w_j

in dual form: prediction $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i (\mathbf{x}_i \cdot \mathbf{x}) + b$

- the contribution of each **instance** is weighted by α_i

Derivation of Linear Support Vector Machines

Dual variables

primal form of linear SVMs can be rewritten with dual variables $\alpha_i \geq 0$ s.t.

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i t_i \mathbf{x}_i \quad \text{where} \quad \sum_{i=1}^{n} \alpha_i t_i = 0$$

$\alpha_i =$ "weight of instance $\mathbf{x}_i$ in expression of $\mathbf{w}$" ($\times \pm 1$ if class $t_i = \pm 1$)

Primal variables vs. dual variables

in primal form: prediction $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{j=1}^{d} w_j x_j + b$

- the contribution of each **feature** is weighted by w_j

in dual form: prediction $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i (\mathbf{x}_i \cdot \mathbf{x}) + b$

- the contribution of each **instance** is weighted by α_i

Derivation of Linear Support Vector Machines

Dual variables

primal form of linear SVMs can be rewritten with dual variables $\alpha_i \geq 0$ s.t.

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i t_i \mathbf{x}_i \quad \text{where} \quad \sum_{i=1}^{n} \alpha_i t_i = 0$$

$\alpha_i =$ "weight of instance $\mathbf{x}_i$ in expression of $\mathbf{w}$" ($\times \pm 1$ if class $t_i = \pm 1$)

Primal variables vs. dual variables

in primal form: prediction $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{j=1}^{d} w_j x_j + b$

- the contribution of each **feature** is weighted by w_j

in dual form: prediction $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i (\mathbf{x}_i \cdot \mathbf{x}) + b$

- the contribution of each **instance** is weighted by α_i

Derivation of Linear Support Vector Machines

Dual variables

primal form of linear SVMs can be rewritten with dual variables $\alpha_i \geq 0$ s.t.

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i t_i \mathbf{x}_i \quad \text{where} \quad \sum_{i=1}^{n} \alpha_i t_i = 0$$

α_i = "weight of instance $\mathbf{x}_i$ in expression of $\mathbf{w}$" ($\times \pm 1$ if class $t_i = \pm 1$)

Primal variables vs. dual variables

in primal form: prediction $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{j=1}^{d} w_j x_j + b$

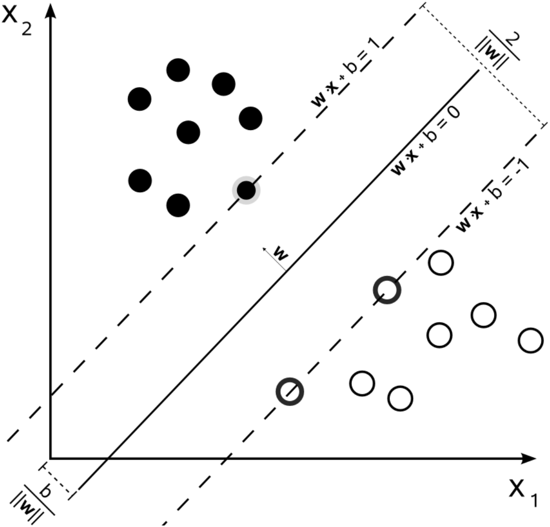
- the contribution of each **feature** is weighted by w_j

in dual form: prediction $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i (\mathbf{x}_i \cdot \mathbf{x}) + b$

- the contribution of each **instance** is weighted by α_i

Support Vectors and Dual Variables

dual variable α_i is non-zero **only** for support vectors s.t. $t_i\,(\mathbf{w}\cdot\mathbf{x}_i + b) = 1$



http://en.wikipedia.org/wiki/Support_vector_machine

Optimisation for Linear Support Vector Machines

Dual form of linear SVMs for separable data

for linearly separable data, maximising the margin is equivalent to solve

$$\begin{aligned} \max \quad & \textstyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \\ \text{s.t.} \quad & \alpha_i \geq 0 \quad \text{and} \quad \textstyle\sum_{i=1}^{n} \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i \left(\mathbf{x}_i \cdot \mathbf{x}\right) + b$

Solving the SVM problem

dual form of SVMs is a quadratic optimisation problem

- convex objective function $\Rightarrow$ no local minima, guaranteed convergence
- very efficient solvers exist to find α_i (e.g. LIBSVM and LIBLINEAR)
- usually, many dual variables are zero (non-SVs) $\Rightarrow$ sparse model
- bounds on the generalisation error of SVMs can be obtained

Optimisation for Linear Support Vector Machines

Dual form of linear SVMs for separable data

for linearly separable data, maximising the margin is equivalent to solve

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j \right) \\ \text{s.t.} \quad & \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i \left(\mathbf{x}_i \cdot \mathbf{x} \right) + b$

Solving the SVM problem

dual form of SVMs is a quadratic optimisation problem

- convex objective function $\Rightarrow$ no local minima, guaranteed convergence
- very efficient solvers exist to find α_i (e.g. LIBSVM and LIBLINEAR)
- usually, many dual variables are zero (non-SVs) $\Rightarrow$ sparse model
- bounds on the generalisation error of SVMs can be obtained

Optimisation for Linear Support Vector Machines

Dual form of linear SVMs for separable data

for linearly separable data, maximising the margin is equivalent to solve

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j \right) \\ \text{s.t.} \quad & \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i \left(\mathbf{x}_i \cdot \mathbf{x} \right) + b$

Solving the SVM problem

dual form of SVMs is a quadratic optimisation problem

- convex objective function $\Rightarrow$ no local minima, guaranteed convergence
- very efficient solvers exist to find α_i (e.g. LIBSVM and LIBLINEAR)
- usually, many dual variables are zero (non-SVs) $\Rightarrow$ sparse model
- bounds on the generalisation error of SVMs can be obtained

Optimisation for Linear Support Vector Machines

Dual form of linear SVMs for separable data

for linearly separable data, maximising the margin is equivalent to solve

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j \right) \\ \text{s.t.} \quad & \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i \left(\mathbf{x}_i \cdot \mathbf{x} \right) + b$

Solving the SVM problem

dual form of SVMs is a quadratic optimisation problem

- convex objective function $\Rightarrow$ no local minima, guaranteed convergence
- very efficient solvers exist to find α_i (e.g. LIBSVM and LIBLINEAR)
- usually, many dual variables are zero (non-SVs) $\Rightarrow$ sparse model
- bounds on the generalisation error of SVMs can be obtained

Optimisation for Linear Support Vector Machines

Dual form of linear SVMs for separable data

for linearly separable data, maximising the margin is equivalent to solve

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j \right) \\ \text{s.t.} \quad & \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i \left(\mathbf{x}_i \cdot \mathbf{x} \right) + b$

Solving the SVM problem

dual form of SVMs is a quadratic optimisation problem

- convex objective function $\Rightarrow$ no local minima, guaranteed convergence
- very efficient solvers exist to find α_i (e.g. LIBSVM and LIBLINEAR)
- usually, many dual variables are zero (non-SVs) $\Rightarrow$ sparse model
- bounds on the generalisation error of SVMs can be obtained

Optimisation for Linear Support Vector Machines

Dual form of linear SVMs for separable data

for linearly separable data, maximising the margin is equivalent to solve

$$\begin{array}{ll} \max & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \\ \text{s.t.} & \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i t_i = 0 \end{array}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i \left(\mathbf{x}_i \cdot \mathbf{x}\right) + b$

Solving the SVM problem

dual form of SVMs is a quadratic optimisation problem

- convex objective function $\Rightarrow$ no local minima, guaranteed convergence
- very efficient solvers exist to find α_i (e.g. LIBSVM and LIBLINEAR)
- usually, many dual variables are zero (non-SVs) $\Rightarrow$ sparse model
- bounds on the generalisation error of SVMs can be obtained

More about Linear Support Vector Machines

bias term can be computed as

$$b = \frac{1}{n} \sum_{j=1}^{n} \left(t_i - \sum_{i=1}^{n} \alpha_i t_i \left(\mathsf{x}_i \cdot \mathsf{x}_j \right) \right)$$

these slides only give a "light" overview of SVM theory

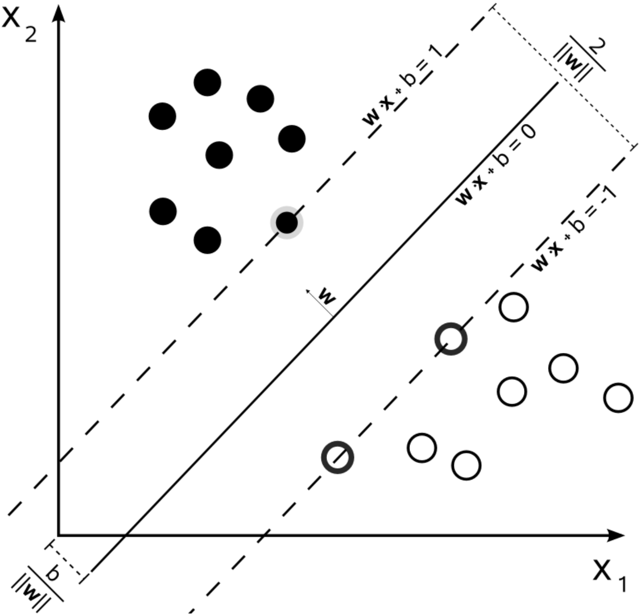
- everything can be formally derived with Lagrangian approach
- many tutorials, see e.g. `http://www.kernel-machines.org`
- "*A Tutorial on Support Vector Machines for Pattern Recognition*"

excellent course about SVMs by Colin Campbell (in two parts)

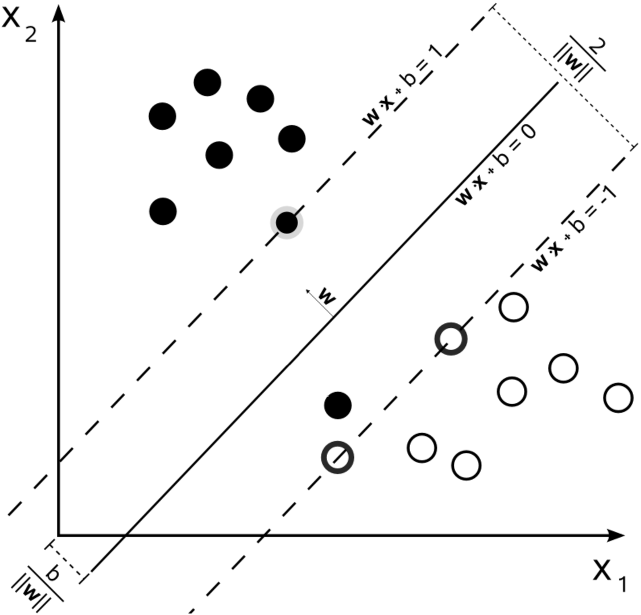
- `http://videolectures.net/epsrcws08_campbell_isvm`

Linear SVMs for Non-Separable Data

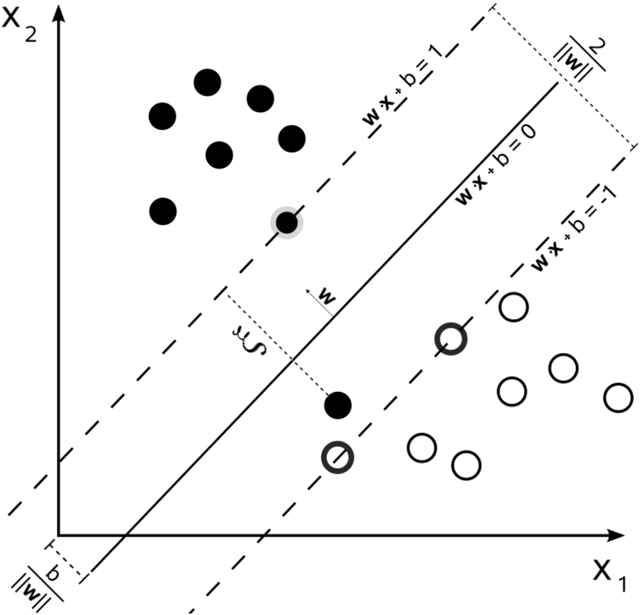
Dealing with Non-Separable Data

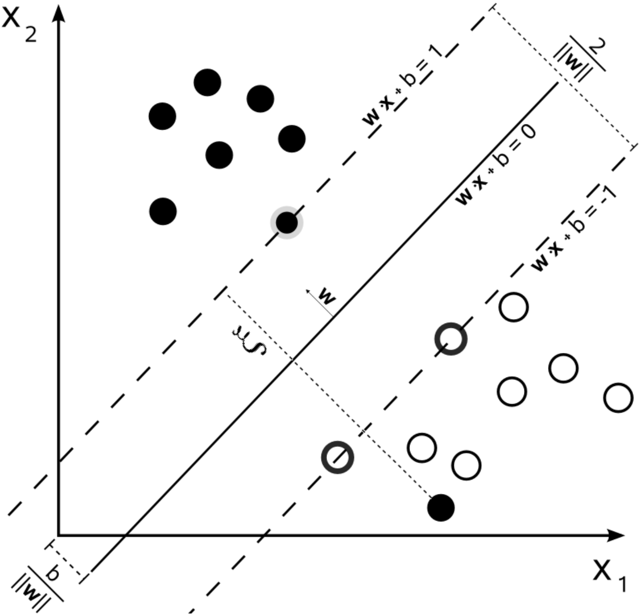


Dealing with Non-Separable Data



Dealing with Non-Separable Data





x_2
x_1
$\mathbf{w} \cdot \mathbf{x} + b = 1$
$\mathbf{w} \cdot \mathbf{x} + b = 0$
$\mathbf{w} \cdot \mathbf{x} + b = -1$
$\frac{2}{\|\mathbf{w}\|}$
$\frac{b}{\|\mathbf{w}\|}$
$\mathbf{w}$
ξ

Linear SVMs with Slack Variables

Primal form of linear SVMs for non-separable data

for linearly non-separable data, two opposite objectives are competing

- maximising the margin for better generalisation
- minimising the sum of slack variables to avoid errors

solution = compromise to achieve a large margin with not too large errors

Primal form of linear SVMs for non-separable data

for linearly non-separable data, the solution is given by

$$\begin{aligned} \min \quad & \|\mathbf{w}\| + C \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & t_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) \geq 1 - \xi_i \end{aligned}$$

where C determines the compromise between model complexity (large $\|\mathbf{w}\|$ = complex model) and prediction errors (large $\sum_{i=1}^{n} \xi_i$ = error-prone)

Linear SVMs with Slack Variables

Primal form of linear SVMs for non-separable data

for linearly non-separable data, two opposite objectives are competing

- maximising the margin for better generalisation
- minimising the sum of slack variables to avoid errors

solution = compromise to achieve a large margin with not too large errors

Primal form of linear SVMs for non-separable data

for linearly non-separable data, the solution is given by

$$\begin{aligned} \min \quad & \|\mathbf{w}\| + C \textstyle\sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & t_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) \geq 1 - \xi_i \end{aligned}$$

where C determines the compromise between model complexity (large $\|\mathbf{w}\|$ = complex model) and prediction errors (large $\sum_{i=1}^{n} \xi_i$ = error-prone)

Linear SVMs with Slack Variables

Primal form of linear SVMs for non-separable data

for linearly non-separable data, two opposite objectives are competing

- maximising the margin for better generalisation
- minimising the sum of slack variables to avoid errors

solution = compromise to achieve a large margin with not too large errors

Primal form of linear SVMs for non-separable data

for linearly non-separable data, the solution is given by

$$\begin{aligned} \min \quad & \|\mathbf{w}\| + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & t_i\left(\mathbf{w}\cdot\mathbf{x}_i + b\right) \geq 1 - \xi_i \end{aligned}$$

where C determines the compromise between model complexity (large $\|\mathbf{w}\|$ = complex model) and prediction errors (large $\sum_{i=1}^{n}\xi_i$ = error-prone)

Linear SVMs with Slack Variables

Primal form of linear SVMs for non-separable data

for linearly non-separable data, two opposite objectives are competing

- maximising the margin for better generalisation
- minimising the sum of slack variables to avoid errors

solution = compromise to achieve a large margin with not too large errors

Primal form of linear SVMs for non-separable data

for linearly non-separable data, the solution is given by

$$\begin{aligned} \min \quad & \|\mathbf{w}\| + C \textstyle\sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & t_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) \geq 1 - \xi_i \end{aligned}$$

where C determines the compromise between model complexity (large $\|\mathbf{w}\|$ = complex model) and prediction errors (large $\sum_{i=1}^{n} \xi_i$ = error-prone)

Linear SVMs with Slack Variables

Primal form of linear SVMs for non-separable data

for linearly non-separable data, two opposite objectives are competing

- maximising the margin for better generalisation
- minimising the sum of slack variables to avoid errors

solution = compromise to achieve a large margin with not too large errors

Primal form of linear SVMs for non-separable data

for linearly non-separable data, the solution is given by

$$\begin{aligned} \text{min} \quad & \|\mathbf{w}\| + C \textstyle\sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & t_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) \geq 1 - \xi_i \end{aligned}$$

where C determines the compromise between model complexity (large $\|\mathbf{w}\|$ = complex model) and prediction errors (large $\sum_{i=1}^{n} \xi_i$ = error-prone)

Linear SVMs with Slack Variables

Dual form of linear SVMs for non-separable data

for linearly non-separable data, the solution is given by

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j \right) \\ \text{s.t.} \quad & C \geq \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i \left(\mathbf{x}_i \cdot \mathbf{x} \right) + b$

the **only difference** with the separable case is the set of constraints $\alpha_i \leq C$

Dealing with non-separable data

- most low-dimensional datasets are linearly non-separable
- if $d \gg n$, the regularisation constant C allows avoiding overfitting
- SVM implementations support penalties $\sum_{i=1}^{n} \xi_i$ and $\sum_{i=1}^{n} \xi_i^2$

Dual form of linear SVMs for non-separable data

for linearly non-separable data, the solution is given by

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \\ \text{s.t.} \quad & C \geq \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i \left(\mathbf{x}_i \cdot \mathbf{x}\right) + b$

the **only difference** with the separable case is the set of constraints $\alpha_i \leq C$

Dealing with non-separable data

- most low-dimensional datasets are linearly non-separable
- if $d \gg n$, the regularisation constant C allows avoiding overfitting
- SVM implementations support penalties $\sum_{i=1}^{n} \xi_i$ and $\sum_{i=1}^{n} \xi_i^2$

Linear SVMs with Slack Variables

Dual form of linear SVMs for non-separable data

for linearly non-separable data, the solution is given by

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ \text{s.t.} \quad & C \geq \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i (\mathbf{x}_i \cdot \mathbf{x}) + b$

the **only difference** with the separable case is the set of constraints $\alpha_i \leq C$

Dealing with non-separable data

- most low-dimensional datasets are linearly non-separable
- if $d \gg n$, the regularisation constant C allows avoiding overfitting
- SVM implementations support penalties $\sum_{i=1}^{n} \xi_i$ and $\sum_{i=1}^{n} \xi_i^2$

Linear SVMs with Slack Variables

Dual form of linear SVMs for non-separable data

for linearly non-separable data, the solution is given by

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \\ \text{s.t.} \quad & C \geq \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i \left(\mathbf{x}_i \cdot \mathbf{x}\right) + b$

the **only difference** with the separable case is the set of constraints $\alpha_i \leq C$

Dealing with non-separable data

- most low-dimensional datasets are linearly non-separable
- if $d \gg n$, the regularisation constant C allows avoiding overfitting
- SVM implementations support penalties $\sum_{i=1}^{n} \xi_i$ and $\sum_{i=1}^{n} \xi_i^2$

Linear SVMs with Slack Variables

Dual form of linear SVMs for non-separable data

for linearly non-separable data, the solution is given by

$$\max \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j \right)$$

$$\text{s.t.} \quad C \geq \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i t_i = 0$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i \left(\mathbf{x}_i \cdot \mathbf{x} \right) + b$

the **only difference** with the separable case is the set of constraints $\alpha_i \leq C$

Dealing with non-separable data

- most low-dimensional datasets are linearly non-separable
- if $d \gg n$, the regularisation constant C allows avoiding overfitting
- SVM implementations support penalties $\sum_{i=1}^{n} \xi_i$ and $\sum_{i=1}^{n} \xi_i^2$

Kernels and Similarity Measures

Taking Advantage of the Dot Products

Dot products in SVM formulation

instances only appear through **dot products** in SVM dual

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j \right) \\ \text{s.t.} \quad & C \geq \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i \left(\mathbf{x}_i \cdot \mathbf{x} \right) + b$

From dot products to kernels

dot product $\mathbf{x}_i \cdot \mathbf{x}_j$ = similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$

- can we use other similarity measures to go non-linear ?
- can we define similarity measures for non-numerical data ?

short answer: yes, we can, if we replace the dot product by a kernel !

Taking Advantage of the Dot Products

Dot products in SVM formulation

instances only appear through **dot products** in SVM dual

$$\begin{aligned} \max \quad & \textstyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j \right) \\ \text{s.t.} \quad & C \geq \alpha_i \geq 0 \quad \text{and} \quad \textstyle\sum_{i=1}^{n} \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i \left(\mathbf{x}_i \cdot \mathbf{x} \right) + b$

From dot products to kernels

dot product $\mathbf{x}_i \cdot \mathbf{x}_j$ = similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$

- can we use other similarity measures to go non-linear ?
- can we define similarity measures for non-numerical data ?

short answer: yes, we can, if we replace the dot product by a kernel !

Taking Advantage of the Dot Products

Dot products in SVM formulation

instances only appear through **dot products** in SVM dual

$$\begin{aligned} \max \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ \text{s.t.} \quad & C \geq \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^n \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^n \alpha_i t_i (\mathbf{x}_i \cdot \mathbf{x}) + b$

From dot products to kernels

dot product $\mathbf{x}_i \cdot \mathbf{x}_j$ = similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$

- can we use other similarity measures to go non-linear ?
- can we define similarity measures for non-numerical data ?

short answer: yes, we can, if we replace the dot product by a kernel !

Taking Advantage of the Dot Products

Dot products in SVM formulation

instances only appear through **dot products** in SVM dual

$$\begin{aligned} \max \quad & \textstyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j \right) \\ \text{s.t.} \quad & C \geq \alpha_i \geq 0 \quad \text{and} \quad \textstyle\sum_{i=1}^{n} \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i \left(\mathbf{x}_i \cdot \mathbf{x} \right) + b$

From dot products to kernels

dot product $\mathbf{x}_i \cdot \mathbf{x}_j$ = similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$

- can we use other similarity measures to go non-linear ?
- can we define similarity measures for non-numerical data ?

short answer: yes, we can, if we replace the dot product by a kernel !

Taking Advantage of the Dot Products

Dot products in SVM formulation

instances only appear through **dot products** in SVM dual

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \\ \text{s.t.} \quad & C \geq \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i \left(\mathbf{x}_i \cdot \mathbf{x}\right) + b$

From dot products to kernels

dot product $\mathbf{x}_i \cdot \mathbf{x}_j$ = similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$

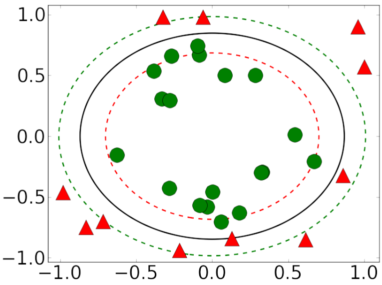
- can we use other similarity measures to go non-linear ?
- can we define similarity measures for non-numerical data ?

short answer: yes, we can, if we replace the dot product by a kernel !

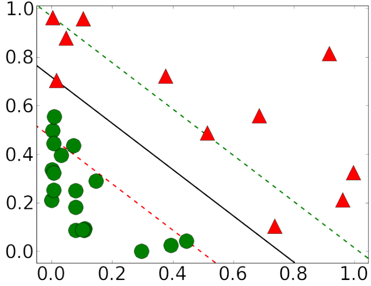
Kernels: the Feature Space Interpretation

SVMs are linear, but most real problems are non-linear !

- use ϕ to map data in a **high-dimensional feature space**
- compute the dot product in the feature space
- kernel function is defined as $k(x, y) = \langle \phi(x), \phi(y) \rangle$



$\Leftrightarrow$



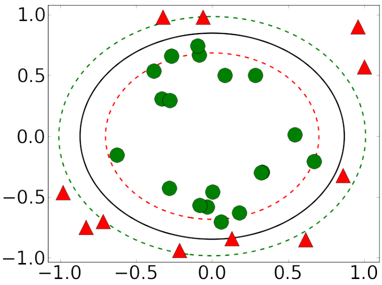
x vs. $y \leftrightarrow$???

x^2 vs. $y^2 \leftrightarrow x^2 + y^2 \leq r^2$

Kernels: the Feature Space Interpretation

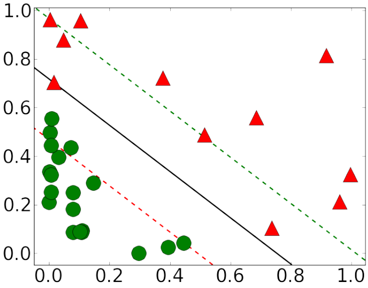
SVMs are linear, but most real problems are non-linear !

- use ϕ to map data in a **high-dimensional feature space**
- compute the dot product in the feature space
- kernel function is defined as $k(x, y) = \langle \phi(x), \phi(y) \rangle$



x vs. $y \leftrightarrow$???

$\Leftrightarrow$

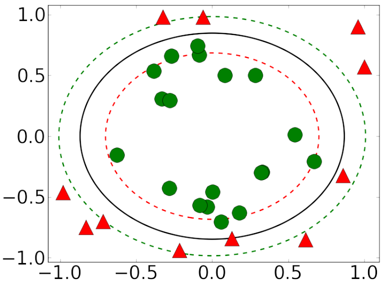


x^2 vs. $y^2 \leftrightarrow x^2 + y^2 \leq r^2$

Kernels: the Feature Space Interpretation

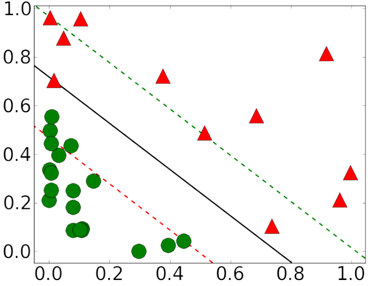
SVMs are linear, but most real problems are non-linear !

- use ϕ to map data in a **high-dimensional feature space**
- compute the dot product in the feature space
- kernel function is defined as $k(x, y) = \langle \phi(x), \phi(y) \rangle$



x vs. $y \leftrightarrow$???

$\Leftrightarrow$

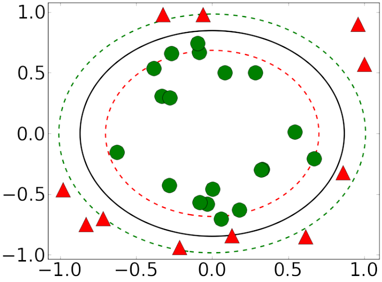


x^2 vs. $y^2 \leftrightarrow x^2 + y^2 \leq r^2$

Kernels: the Feature Space Interpretation

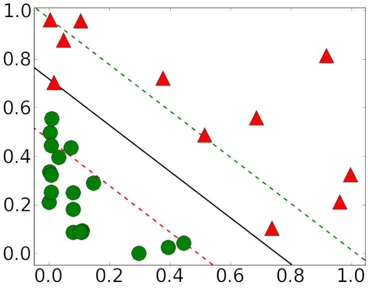
SVMs are linear, but most real problems are non-linear !

- use ϕ to map data in a **high-dimensional feature space**
- compute the dot product in the feature space
- kernel function is defined as $k(x, y) = \langle \phi(x), \phi(y) \rangle$



x vs. $y \leftrightarrow$???

$\Leftrightarrow$



x^2 vs. $y^2 \leftrightarrow x^2 + y^2 \leq r^2$

Kernels: the Similarity Measure Interpretation

From dot products to kernels

in practice, the RBF kernel is very often used

$$k(x, z) = \exp(-\gamma \|x - z\|^2) \in [0, 1]$$

and can be interpreted as a similarity measure between x and z

Kernels for non-numerical data

kernels can be used to compare DNA sequences, graphs, images...

- kernels are symmetric, i.e. $k(x, z) = k(z, x)$
- kernels can be combined (sum, product, etc.)

the Gram matrix $\mathbf{K}$ s.t. $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ should be positive semi-definite

Kernels: the Similarity Measure Interpretation

From dot products to kernels

in practice, the RBF kernel is very often used

$$k(x, z) = \exp(-\gamma\|x - z\|^2) \in [0, 1]$$

and can be interpreted as a similarity measure between x and z

Kernels for non-numerical data

kernels can be used to compare DNA sequences, graphs, images...

- kernels are symmetric, i.e. $k(x, z) = k(z, x)$
- kernels can be combined (sum, product, etc.)

the Gram matrix $\mathbf{K}$ s.t. $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ should be positive semi-definite

Kernelised SVMs for Non-Linear Data

Kernelised SVM Formulation

Dual form of kernelised SVMs for non-separable data

for non-linearly non-separable data, the solution is given by

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k\left(\mathbf{x}_i, \mathbf{x}_j\right) \\ \text{s.t.} \quad & C \geq \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i t_i = 0 \end{aligned}$$

and to make predictions with $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i t_i k\left(\mathbf{x}_i, \mathbf{x}\right) + b$

Kernel parameters

SVM complexity also depends on the kernel parameter, e.g. for RBF kernel

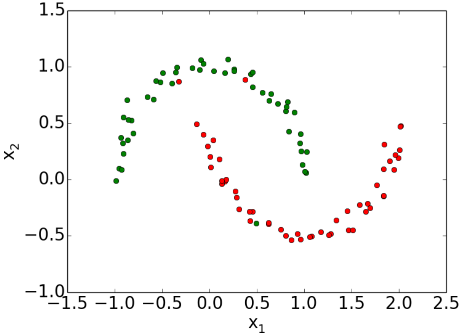
$$k(x, z) = \exp(-\gamma \|x - z\|^2) \in [0, 1]$$

- small parameter γ = focus on large structures
- large parameter γ = focus on small details

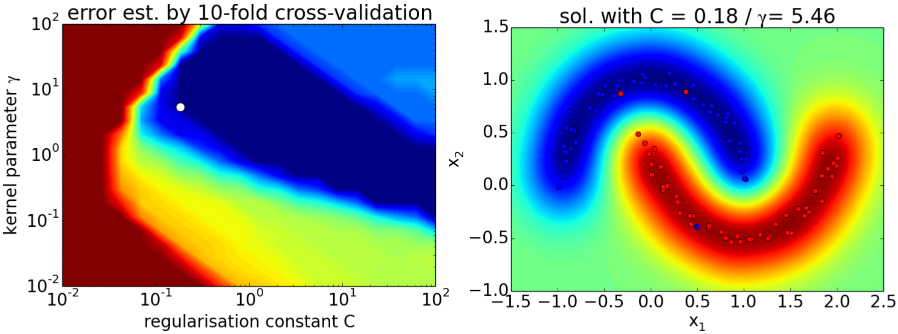
Experimental Settings

Dataset

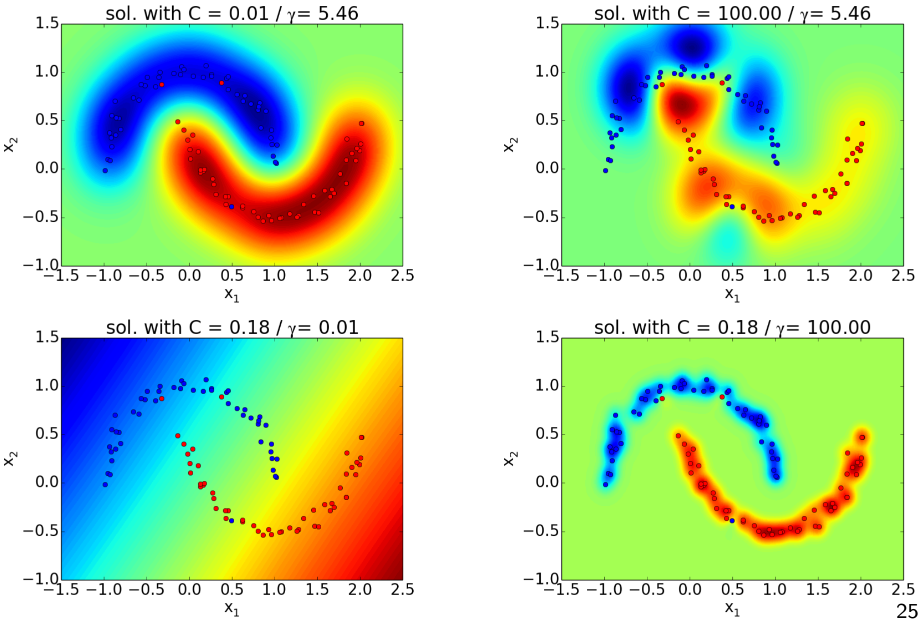
- artificial problem "Two Moons" (`sklearn.datasets.make_moons`)
- $n = 30$ (inc. 3 mislabelled) with non-linear support vector machine
  - meta-parameter C : regularisation constant (simple $\leftrightarrow$ complex)
  - meta-parameter γ: scale at which we "look" at data (small $\leftrightarrow$ large)



Results with 10-fold Cross-Validation



Results with Suboptimal Meta-parameter Choices



Going Further with Kernels

Can we Use Kernels Only with SVMs ?

Short answer

any method where instances appear through dot products can be kernelised

More details

many methods have been kernelised (non-exhaustive list):

- linear regression
- logistic regression
- principal component analysis
- clustering techniques

warning: complexity **must be controlled** by regularisation techniques

Softwares for Kernel Machines

LIBSVM

supports kernels, most frequently used implementation, many wrappers

- http://www.csie.ntu.edu.tw/~cjlin/libsvm/

LIBLINEAR

very efficient implementation of linear SVMs, much faster, many wrappers

- http://www.csie.ntu.edu.tw/~cjlin/liblinear/

scikit-learn

provides an interface to LIBSVM from Python

- http://scikit-learn.org/stable/modules/svm.html
- http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Outline of this Lesson

- maximum margin classifiers
- linear SVMs for separable data
- linear SVMs for non-separable data
- kernels and similarity measures
- kernelised SVMs for non-linear data
- going further with kernels

References

many tutorials, see e.g. `http://www.kernel-machines.org`