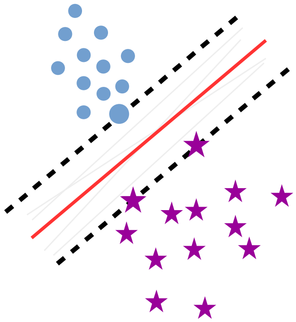
Introduction to AI
Support Vector Machines



Rodrigo Cabral

EUR DS4H-LIFE-SPECTRUM

cabral@unice.fr

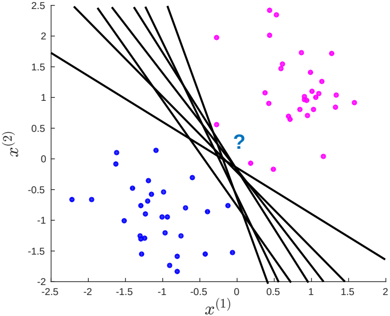
Outline

1. Hard-margin support vector machines for binary classification
2. Soft-margin support vector machines for binary classification
3. Nonlinear support vector machines
4. Multi-class classification
5. Conclusions

Linearly separable classes

What happens if the classes are separable?

- Which line do you choose?



- Logistic regression suffers from two major issues in this case:
  1. $\kappa \to +\infty$ to have $J \to 0$: optimization algorithms become unstable.
  2. The solution of the problem is not unique.

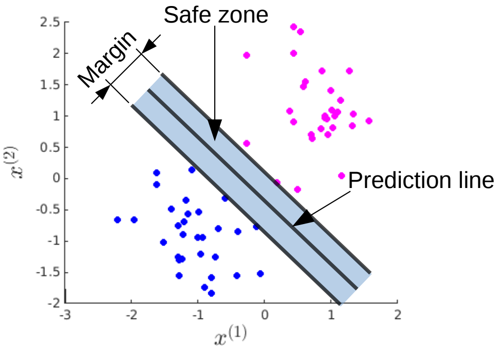
Are there other approaches different from regularized logistic regression?

Large margin linear decisions

What would be a good decision line?

- Line with a “safe zone” around it.
  - Correct classification for new data points which can located around training data.
  - Good generalization properties.

- Margin: maximum width around the decision line before hitting a data point.

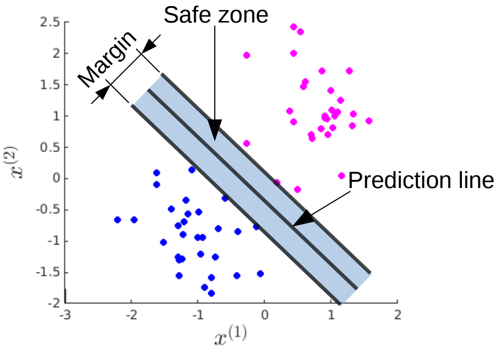


Large margin linear decisions

What would be a good decision line?

- Line with a “safe zone” around it.
  - Correct classification for new data points which can located around training data.
  - Good generalization properties.

- Margin: maximum width around the decision line before hitting a data point.
- Choose decision line with **largest margin**.

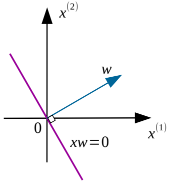


Reminder: hyperplane equations and half-spaces

Equation of an hyperplane passing through 0

$\mathbf{x}\mathbf{w} = 0$

$\|\mathbf{w}\|_2 = 1$ - unitary vector

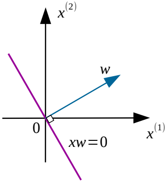


Reminder: hyperplane equations and half-spaces

Equation of an hyperplane passing through 0

$\mathbf{x}\mathbf{w} = 0$

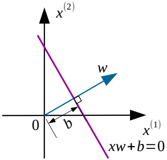
$\|\mathbf{w}\|_2 = 1$ - unitary vector



General hyperplane equation

$\mathbf{x}\mathbf{w} + b = 0$

b - signed distance of closest point to origin

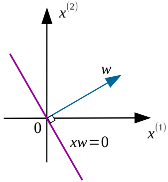


Reminder: hyperplane equations and half-spaces

Equation of an hyperplane passing through 0

$\mathbf{x}\mathbf{w} = 0$

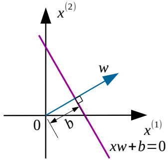
$\|\mathbf{w}\|_2 = 1$ - unitary vector



General hyperplane equation

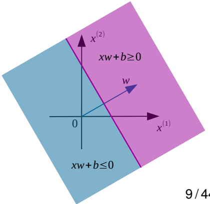
$\mathbf{x}\mathbf{w} + b = 0$

b - signed distance of closest point to origin

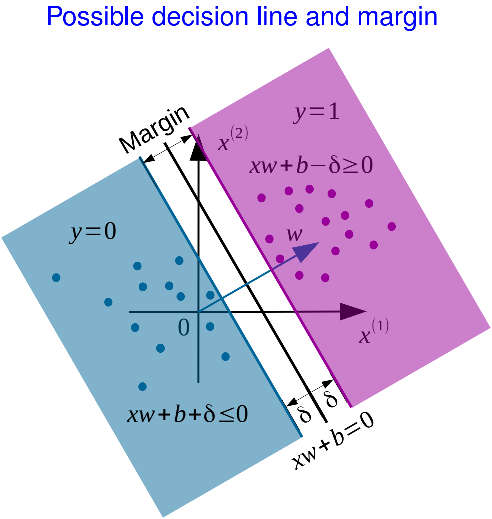


Upper half-space $\qquad \mathbf{x}\mathbf{w} + b \geq 0$

Lower half-space $\qquad \mathbf{x}\mathbf{w} + b \leq 0$



Large margin linear decisions



- **Objective:** choose line parameters **w** and b leading to maximum margin 2δ such that data points are outside the margin.

Large margin linear decisions

Problem formulation as constrained minimization

- The margin constraint for a data point $\mathbf{x}_i$ is

$$\begin{cases} \mathbf{x}_i \mathbf{w} + b \geq \delta & \text{if} \quad y_i = 1 \\ \mathbf{x}_i \mathbf{w} + b \leq -\delta & \text{if} \quad y_i = 0 \end{cases}$$

- Dividing both sides by δ, defining $\boldsymbol{\beta}' = \mathbf{w}/\delta$ and $\beta_0 = b/\delta$, we have

$$\begin{cases} \mathbf{x}_i \boldsymbol{\beta}' + \beta_0 \geq 1 & \text{if} \quad y = 1 \\ \mathbf{x}_i \boldsymbol{\beta}' + \beta_0 \leq -1 & \text{if} \quad y = 0 \end{cases}$$

- This can be rewritten in a more compact form as follows

$$(2y_i - 1)\left(\mathbf{x}_i \boldsymbol{\beta}' + \beta_0\right) \geq 1$$

Large margin linear decisions

Problem formulation as constrained minimization

- Note that $\delta = 1/\|\boldsymbol{\beta}'\|_2$, therefore the optimization problem we want to solve is

$$
\begin{array}{cc}
\text{maximize} & \frac{2}{\|\boldsymbol{\beta}'\|_2} \\
\text{with respect to} & \boldsymbol{\beta}',\ \beta_0 \\
\text{subject to} & (2y_i - 1)\left(\mathbf{x}_i \boldsymbol{\beta}' + \beta_0\right) \geq 1 \\
& \text{for all } i \in \{1, \cdots, N\}
\end{array}
$$

Large margin linear decisions

Problem formulation as convex optimization

- The previous problem can be transformed into the following equivalent problem:

$$\begin{array}{cc} \text{minimize} & \frac{\|\boldsymbol{\beta}'\|_2^2}{2} \\ \text{with respect to} & \boldsymbol{\beta}',\ \beta_0 \\ \text{subject to} & (2y_i - 1)\left(\mathbf{x}_i\boldsymbol{\beta}' + \beta_0\right) \geq 1 \\ & \text{for all } i \in \{1, \cdots, N\} \end{array}$$

- This is a convex optimization problem. More precisely a problem from the class of quadratic programs.
- It does not have a closed-form solution.
  $\Longrightarrow$ Fortunately, many numerical optimization algorithms can be used to solve it.

Large margin linear decisions

Dual formulation

- In practice, this optimization problem is very complex to solve. Moreover, the solution cannot be interpreted.
- The problem is recast in its **Langrangian dual form**:

$$
\begin{array}{ll}
\text{maximize} & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j (2y_i - 1)(2y_j - 1) \mathbf{x}_i \mathbf{x}_j^{\mathrm{T}} \\
\text{with respect to} & \alpha_1, \cdots, \alpha_N \\
\text{subject to} & \alpha_1 \geq 0, \cdots, \alpha_N \geq 0 \\
 & \sum_{i=1}^{N} \alpha_i (2y_i - 1) = 0
\end{array}
$$

- α_i are called the Lagrangian dual variables.

Large margin linear decisions

Solution and prediction

- It can be shown that the optimal solution $\hat{\boldsymbol{\beta}}'$, β_0 of the initial problem is then written as a function of the solution of the dual α_i:

$$\hat{\boldsymbol{\beta}}' = \sum_{i=1}^{N} \alpha_i (2y_i - 1)\mathbf{x}_i^{\mathsf{T}}$$

and for any i for which $\alpha_i > 0$ we can retrieve $\hat{\beta}_0$ by solving

$$(2y_i - 1)\left(\mathbf{x}_i \boldsymbol{\beta}' + \beta_0\right) = 1$$

- **Prediction**:

$$\hat{y}(\mathbf{x}) = \begin{cases} 1, & \text{if} \quad f_{\boldsymbol{\beta}}(\mathbf{x}) \geq 0 \\ 0, & \text{if} \quad f_{\boldsymbol{\beta}}(\mathbf{x}) < 0 \end{cases}$$

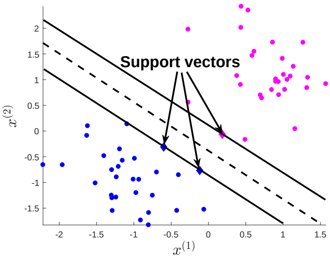
where $f_{\boldsymbol{\beta}}(\mathbf{x}) = \mathbf{x}\hat{\boldsymbol{\beta}}' + \hat{\beta}_0 = \sum_{i=1}^{N} \alpha_i (2y_i - 1)\mathbf{x}\mathbf{x}_i^{\mathsf{T}} + \hat{\beta}_0$.

Support vector machine

Support vectors

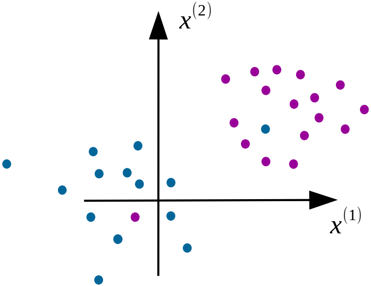
- It can be shown that $\alpha_i > 0$, only if $\mathbf{x}_i$ lies exactly on the optimal margin boundary.
- The prediction line is determined only by these $\mathbf{x}_i$, which are closer to the border.
  $\Longrightarrow$ Decision line is defined only by most ambiguous observations.
- If you remove one of these points from the data set, the decision line may change. These data vectors "support" the decision line.
  $\Longrightarrow$ They are called **support vectors** and the method is called **support vector machine (SVM)**.

Example of linear SVM classifier

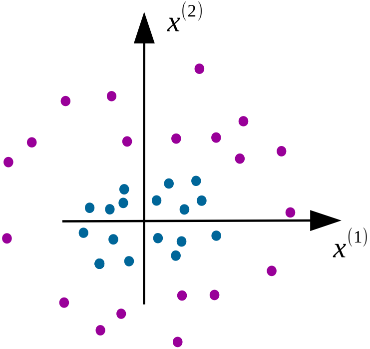


Classes not fully separable

What do we do if the classes are not linearly separable ?



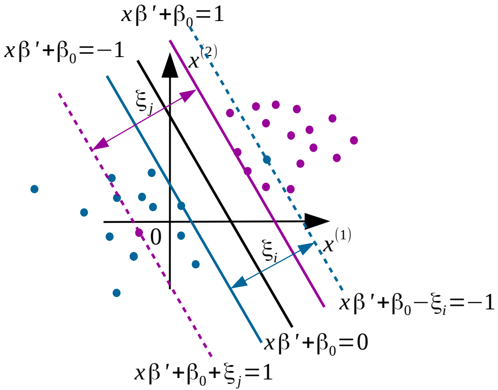
- The data is almost linearly separable, except for a few points.
- We will deal with this case first.



- The data is not close to linear separability, at least with these features.

Soft margin SVM

Soft version of the problem including “slacks” on the margins



- We soften the margins by allowing slacks $\xi_i \geq 0$.
- We should try to keep slacks as small as possible by penalizing them.
  $\Longrightarrow$ Still trying to keep decision line as far as possible from the two classes.

Soft margin SVM

Problem formulation as convex optimization

- We have only slight modifications of the hard margin problem:

$$
\begin{array}{ll}
\text{minimize} & \frac{\|\boldsymbol{\beta}'\|_2^2}{2} + C \sum_{i=1}^{N} \xi_i \\
\text{with respect to} & \boldsymbol{\beta}',\ \beta,\ \xi_1, \cdots, \xi_N \\
\text{subject to} & (2y_i - 1)\left(\mathbf{x}_i \boldsymbol{\beta}' + \beta_0\right) \geq 1 - \xi_i \quad i \in \{1, \cdots, N\} \\
 & \xi_1 \geq 0, \cdots, \xi_N \geq 0
\end{array}
$$

- Coefficient C allows to control trade-off between margin maximization and fitting to data.

Soft margin SVM

Dual formulation

- It can be shown that the dual formulation only changes slightly:

$$\text{maximize} \qquad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j (2y_i - 1)(2y_j - 1) \mathbf{x}_i \mathbf{x}_j^{\mathsf{T}}$$

$$\text{with respect to} \qquad \alpha_1, \cdots, \alpha_N$$

$$\text{subject to} \qquad 0 \leq \alpha_i \leq C \text{ for all } \alpha_i$$

$$\sum_{i=1}^{N} \alpha_i (2y_i - 1) = 0$$

Soft margin SVM

Dual formulation

- It can be shown that the dual formulation only changes slightly:

$$\text{maximize} \qquad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j (2y_i - 1)(2y_j - 1) \mathbf{x}_i \mathbf{x}_j^{\mathsf{T}}$$

$$\text{with respect to} \qquad \alpha_1, \cdots, \alpha_N$$

$$\text{subject to} \qquad 0 \leq \alpha_i \leq C \text{ for all } \alpha_i$$

$$\sum_{i=1}^{N} \alpha_i (2y_i - 1) = 0$$

- We still have

$$\hat{\boldsymbol{\beta}}' = \sum_{i=1}^{N} \alpha_i (2y_i - 1) \mathbf{x}_i^{\mathsf{T}}$$

but $\hat{\beta}_0$ is different:

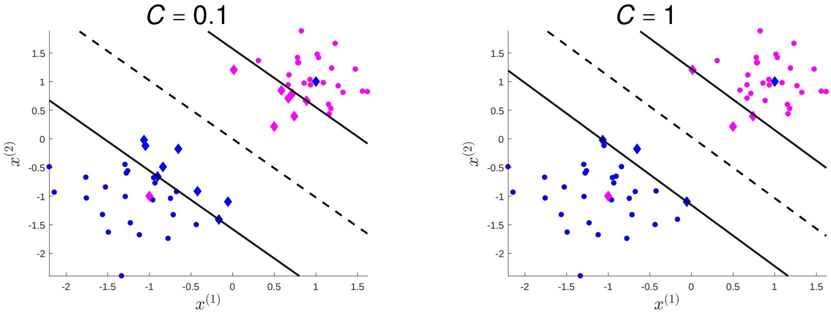
$$\hat{\beta}_0 = (2y_k - 1)(1 - \xi_k) - \mathbf{x}_k \hat{\boldsymbol{\beta}}' \quad \text{with } k = \arg\max_i \alpha_i$$

Soft margin SVM

Support vectors and effect of *C*

- Support vectors are still $\mathbf{x}_i$ corresponding to positive α_i.
- The predictions are given in the same way as before.

Same example but with different *C*



- Support vectors are either on the margin border or beyond it.

1. Hard-margin support vector machines for binary classification

2. Soft-margin support vector machines for binary classification

3. Nonlinear support vector machines

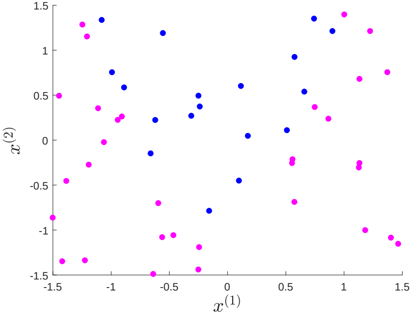
4. Multi-class classification

5. Conclusions

Nonlinear separation

What do we do when classes are not directly linearly separable?

- Example of data set with classes which are not linearly separable in their original features:



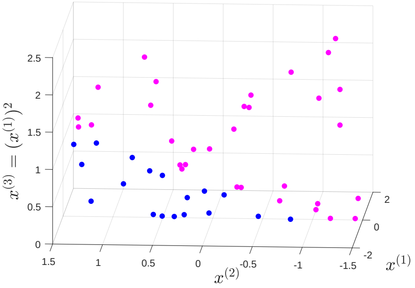
- We can do the same as we did in linear regression to fit nonlinear curves: add transformations of the features.
$\Longrightarrow$The border seems quadratic, we can try adding $\left(x^{(1)}\right)^2$ to the features.

Nonlinear separation with feature transformation

Transformed feature space

- Transformation of each data feature with a function $\phi(\mathbf{x})$ generates an augmented feature space:

$$\mathbf{x}' = \phi(\mathbf{x}) = \begin{bmatrix} x^{(1)} & x^{(2)} & \left(x^{(1)}\right)^2 \end{bmatrix}$$

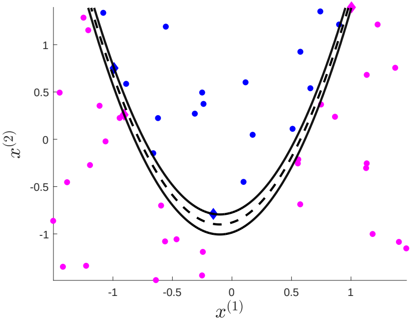


- Classes can be separated by a plane.

$\Longrightarrow$ Linear separation!

SVM with transformed observations

What do we get if we apply SVM with this extended feature space?



- Classes can be perfectly separated with a hard-margin SVM.

SVM with transformed observations

Dual formulation

- For a general transformation $\phi(\mathbf{x}) : \mathbb{R}^M \to \mathbb{R}^L$, the dual formulation of the soft SVM changes to

$$\text{maximize} \qquad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j (2y_i - 1)(2y_j - 1) \phi(\mathbf{x}_i) \phi^{\mathsf{T}}(\mathbf{x}_j)$$

$$\text{with respect to} \qquad \alpha_1, \cdots, \alpha_N$$

$$\text{subject to} \qquad 0 \le \alpha_i \le C \text{ for all } \alpha_i$$

$$\sum_{i=1}^{N} \alpha_i (2y_i - 1) = 0$$

SVM with transformed observations

Dual formulation

- For a general transformation $\phi(\mathbf{x}) : \mathbb{R}^M \to \mathbb{R}^L$, the dual formulation of the soft SVM changes to

$$\text{maximize} \qquad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j (2y_i - 1)(2y_j - 1) \phi(\mathbf{x}_i) \phi^{\mathsf{T}}(\mathbf{x}_j)$$

$$\text{with respect to} \qquad \alpha_1, \cdots, \alpha_N$$

$$\text{subject to} \qquad 0 \le \alpha_i \le C \text{ for all } \alpha_i$$

$$\sum_{i=1}^{N} \alpha_i (2y_i - 1) = 0$$

- The function $f_{\boldsymbol{\beta}}(\mathbf{x})$ used for prediction changes to

$$f_{\boldsymbol{\beta}}(\mathbf{x}) = \mathbf{x}\hat{\boldsymbol{\beta}}' + \hat{\beta}_0 = \sum_{i=1}^{N} \alpha_i (2y_i - 1) \phi(\mathbf{x}) \phi^{\mathsf{T}}(\mathbf{x}_i) + \hat{\beta}_0$$

and $\hat{\beta}_0$ is

$$\hat{\beta}_0 = (2y_k - 1)(1 - \xi_k) - \sum_{i=1}^{N} \alpha_i (2y_i - 1) \phi(\mathbf{x}_k) \phi^{\mathsf{T}}(\mathbf{x}_i) \quad \text{with } k = \arg\max_i \alpha_i$$

Kernels

Kernels

- The transformed features appear always through scalar products

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})\phi^{\mathsf{T}}(\mathbf{x}')$$

- For any chosen $\phi(\cdot)$, $k(\mathbf{x}, \mathbf{x}')$ is a **kernel function**.
- Kernel functions measure similarity between vectors.

Kernels

Kernels

- The transformed features appear always through scalar products

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})\phi^{\mathsf{T}}(\mathbf{x}')$$

- For any chosen $\phi(\cdot)$, $k(\mathbf{x}, \mathbf{x}')$ is a **kernel function**.
- Kernel functions measure similarity between vectors.
- What if we do not know the transformation $\phi(\cdot)$?
  $\Longrightarrow$ Choose directly the kernel function.

Kernels

Kernels

- The transformed features appear always through scalar products

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})\phi^{\mathsf{T}}(\mathbf{x}')$$

- For any chosen $\phi(\cdot)$, $k(\mathbf{x}, \mathbf{x}')$ is a **kernel function.**
- Kernel functions measure similarity between vectors.
- What if we do not know the transformation $\phi(\cdot)$?
  $\Longrightarrow$ Choose directly the kernel function.
- Examples of kernel functions:
  - Radial basis function (RBF): $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|_2^2\right)$
  - Exponential: $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|_2\right)$
  - d-th degree polynomial: $k(\mathbf{x}, \mathbf{x}') = \left(\gamma \mathbf{x}\mathbf{x}'^{\mathsf{T}} + r\right)^d$

Kernel SVM

Dual formulation with kernels - Kernel SVM

- For a kernel $k(\cdot,\cdot)$, the dual formulation of the soft SVM becomes

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j (2y_i - 1)(2y_j - 1) k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{with respect to} & \alpha_1, \cdots, \alpha_N \\ \text{subject to} & 0 \leq \alpha_i \leq C \text{ for all } \alpha_i \\ & \sum_{i=1}^{N} \alpha_i (2y_i - 1) = 0 \end{array}$$

- The function $f_{\boldsymbol{\beta}}(\mathbf{x})$ used for prediction changes to

$$f_{\boldsymbol{\beta}}(\mathbf{x}) = \mathbf{x}\hat{\boldsymbol{\beta}}' + \hat{\beta}_0 = \sum_{i=1}^{N} \alpha_i (2y_i - 1) k(\mathbf{x}, \mathbf{x}_i) + \hat{\beta}_0$$

and $\hat{\beta}_0$ is

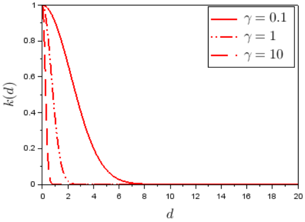
$$\hat{\beta}_0 = (2y_k - 1)(1 - \xi_k) - \sum_{i=1}^{N} \alpha_i (2y_i - 1) k(\mathbf{x}_k, \mathbf{x}_i) \quad \text{with } k = \arg\max_i \alpha_i$$

Kernel SVM

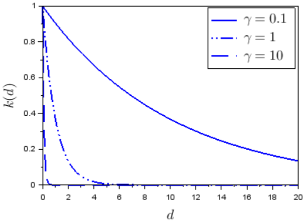
Kernel value as a function of distance

- Let us define the distance $d = \|\mathbf{x} - \mathbf{x}'\|_2$.

RBF: $k(d) = \exp\left(-\gamma d^2\right)$



Exponential: $k(d) = \exp\left(-\gamma d\right)$



- $\gamma > 0$ is an hyperparameter indicating how support vectors will influence decisions on their neighborhoods.

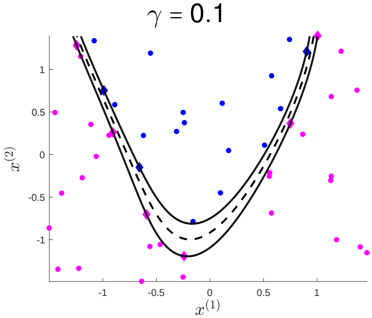
$\implies$ Large γ produces local influence.

$\implies$ Small γ produces global influence.

Kernel SVM

Example

- Previous nonlinear separation example with hard margin kernel SVM: RBF kernel with two different γ.

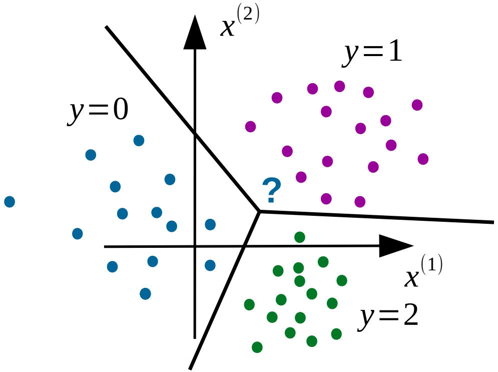


- γ needs to be chosen wisely depending on the complexity of the decision boundary.

Multi-class classification problem

Problem with K classes

- Given classes with K different labels $y \in \{0, \cdots, K-1\}$, how do we define the decision boundaries to generate the predictions?



- Three methods: **One *vs.* one**, **One *vs.* rest**, **Multi-class**.

Multi-class classification problem

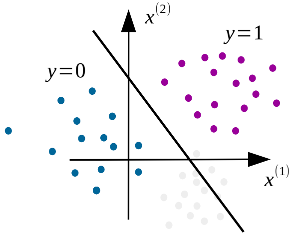
One *vs.* one (OVO)

1. For each pair of classes (k, k'), $k \neq k'$, learn the binary classifier parameters $\beta_{(k,k')}$ to generate predictions $\hat{y}_{(k,k')}$ disregarding the other classes.
2. Combine binary classifiers via voting mechanism, for example, majority voting:

$$\hat{y}_{\text{OVO}}(\mathbf{x}) = \underset{1 \leq k' \leq K}{\arg\max} \left| \left\{ k : \hat{y}_{(k,k')} = k' \right\} \right|$$

where $|\cdot|$ denotes the number of elements of a set.

OVO training



Drawbacks:

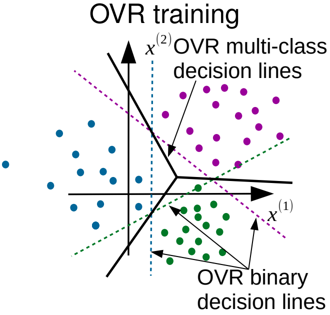
- Computational: train all combinations of binary classifiers.
- Overfitting: size of training sample could be small for a given pair.

Multi-class classification problem

One *vs.* rest (OVR)

1. Learn binary classifier parameters β_k for each class against all the other classes merged.
2. Since $f_{\boldsymbol{\beta}_k}(\mathbf{x})$ is a measure of the depth of $\mathbf{x}$ within class k, prediction is given by

$$\hat{y}_{\mathrm{OVR}}(\mathbf{x}) = \arg\max_{1 \le k \le K} f_{\boldsymbol{\beta}_k}(\mathbf{x})$$



Drawback:

- Calibration: classifier functions $f_{\boldsymbol{\beta}_k}(\mathbf{x})$ may not be comparable.

  It is however quite simple to implement and less complex than OVO.

Multi-class classification problem

Multi-class approach

- Both logistic regression and SVM can be modified to explicitly deal with multiple classes.
  $\Longrightarrow$ No need to learn binary classifiers and to apply fusion rules.

- We are not going to detail this approach in this class.
  $\Longrightarrow$ This kind of approach may increase learning complexity, without leading to a significant increase in classification performance with respect to OVO and OVR.

Conclusions

Logistic regression

- Logistic regression can be seen as a simple adaptation of linear regression to do classification. Learning is equivalent to solve a smooth optimization problem.
- If we want nonlinear boundaries we need to include them explicitly.
- Parameter vector $\hat{\beta}$ allows for interpretation of the importance of the features.
- Complexity of underlying optimization problem depends on the dimension of feature space.
- It is sensible to outliers (it is designed for this) and it cannot be used directly for separable problems.

Conclusions

SVM

- SVM focus directly on the classification problem. It learns a robust separation boundary. Learning is equivalent to solve a quadratic program.
- The boundary can be turned nonlinear either with feature transformation or with kernel SVM.
  $\Longrightarrow$ Kernels can be designed for non-numeric data types (graphs, sequences, relational data).
  $\Longrightarrow$ Kernel SVM has been applied to many different fields ranging from text to genetic data.
- Dual formulation allows to know which observations are important for the SVM classifier, they are the support vectors.

Conclusions

SVM

- In the standard SVM approach, $\hat{\beta}'$ allows to analyze the importance of the features.
  In kernel SVM, there is no $\hat{\beta}'$, therefore the importance of the features is difficult to be analyzed.
- Complexity of underlying optimization problem depends on the number of observations mainly.
- It can be made insensible to outliers (with C) and it can directly deal with separable problems.
- Tuning kernel SVM parameters C, type of kernel and γ can be quite difficult. Most common approach consists in using cross-validation (topic of more advanced lectures).