Kernels

Main points about the Kernel

Main points

- high-dimensional spaces help to separate the data
- project data in high dimensional space: $\mathbf{x} \rightarrow \phi(\mathbf{x})$
- Kernel (by definition) models dot products in the high-dimensional space

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}')$$

- often, we don't know the projection ϕ, but we just need to use the dot product which is given by the kernel

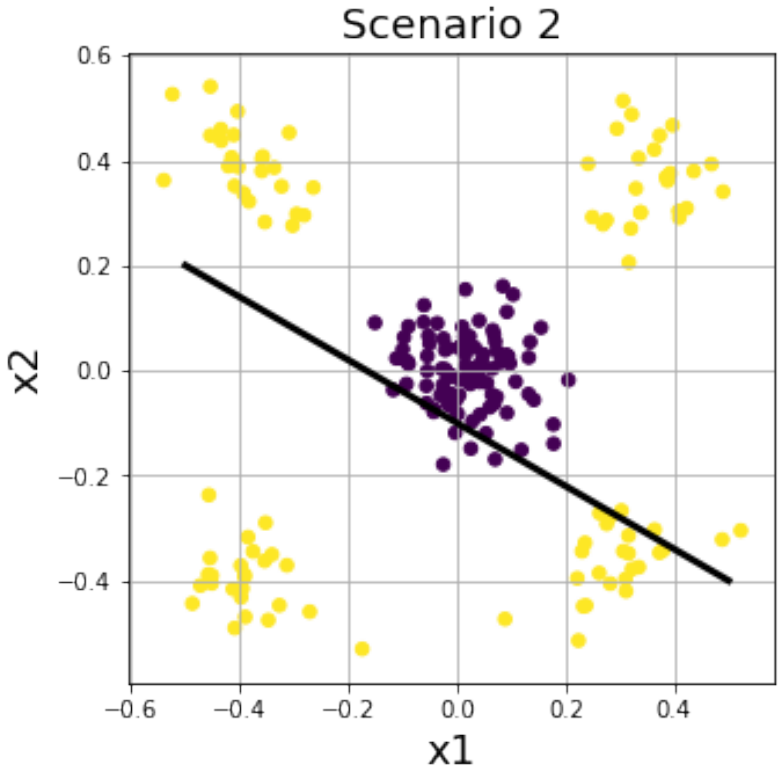
So what do kernels represent?

- how similar the points are, but the notion of similarity may not always match our intuition (cf polynomial kernels)
- classification view point: **points which are 'close' (according to the kernel) should share the same label**

Higher-Dimensional Feature Spaces

2D Synthetic Dataset -Scenario 2-

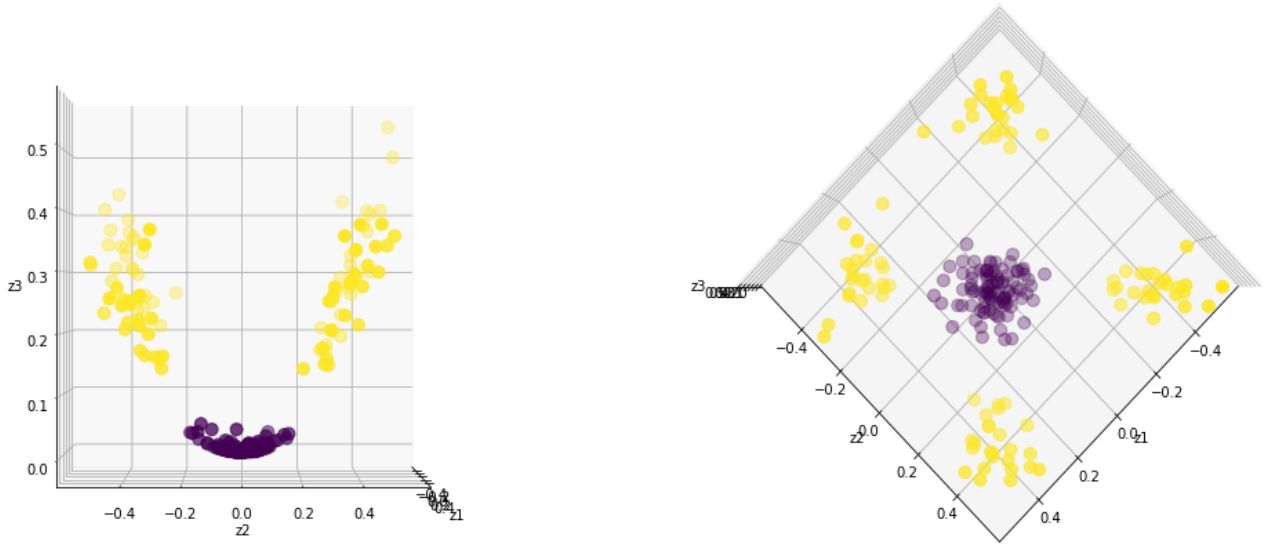
- Non-linearly separable.
- Five clusters.
- 100 samples per class



Higher-Dimensional Feature Spaces

Exercise 1: Adding features

$z_3 = x_1^2 + x_2^2$

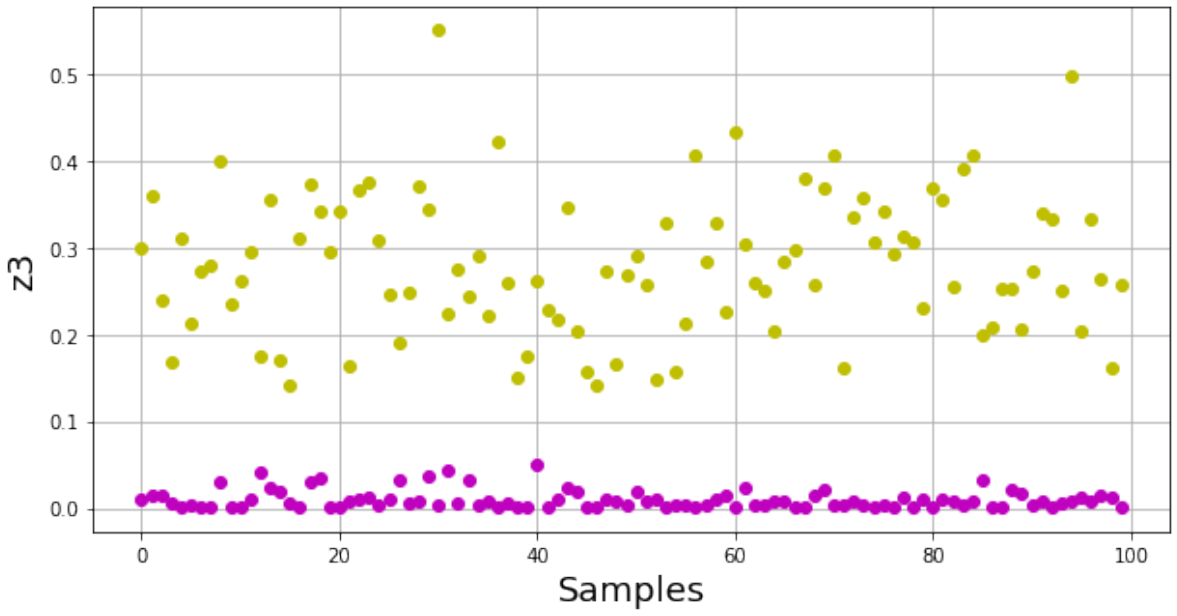


Note that z_3 is the distance to the center point $x_c = [0, 0]$ (purple class)
$(z_3 = (x_1 - 0)^2 + (x_2 - 0)^2)$

Higher-Dimensional Feature Spaces

Exercise 1: Adding features

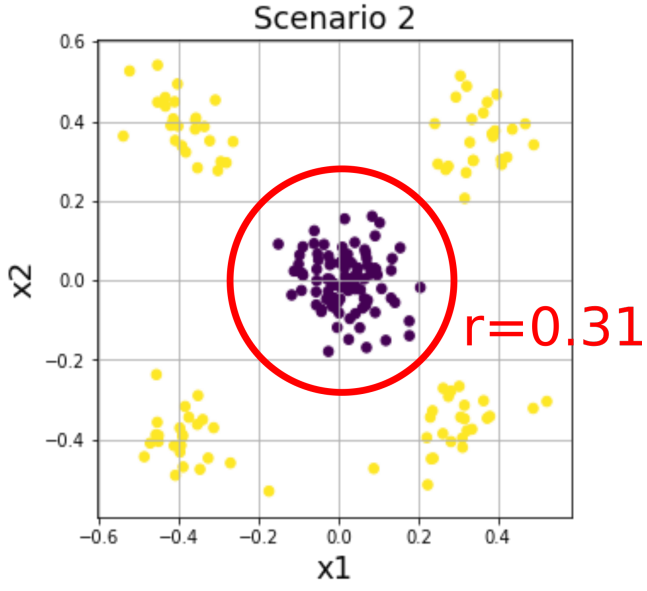
Note also that indeed, using only z_3 both classes can be linearly separated (eg. setting a threshold around 0.1)



Higher-Dimensional Feature Spaces

Exercise 1: Adding features

In this view, any other monotonic function of z_3 could be used to make the class separable, like for instance r^2 below.



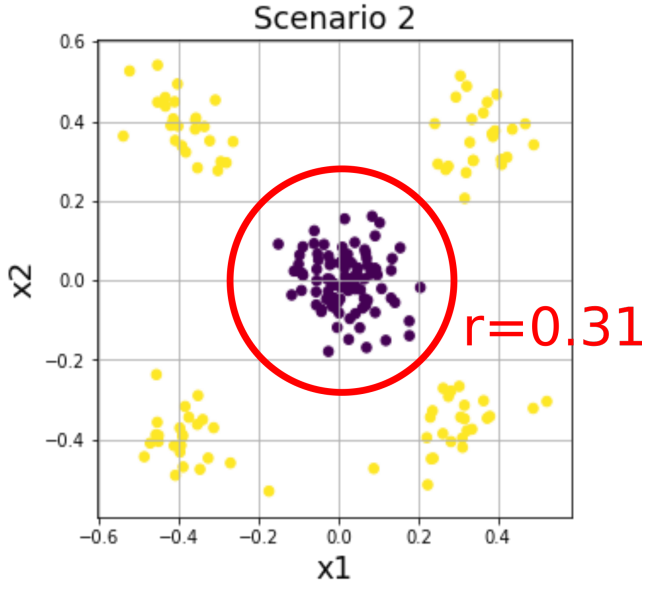
$$z3 = (x1 - 0)^2 + (x2 - 0)^2$$
$$r = \sqrt{z_3}$$

$r_{thresh} = \sqrt{0.1} = 0.31$ is a threshold which could separate the two classes.

Higher-Dimensional Feature Spaces

Exercise 1: Adding features

In this view, any other monotonic function of z_3 could be used to make the class separable, like for instance r^2 below.



$$z3 = (x1 - 0)^2 + (x2 - 0)^2$$
$$r = \sqrt{z_3}$$

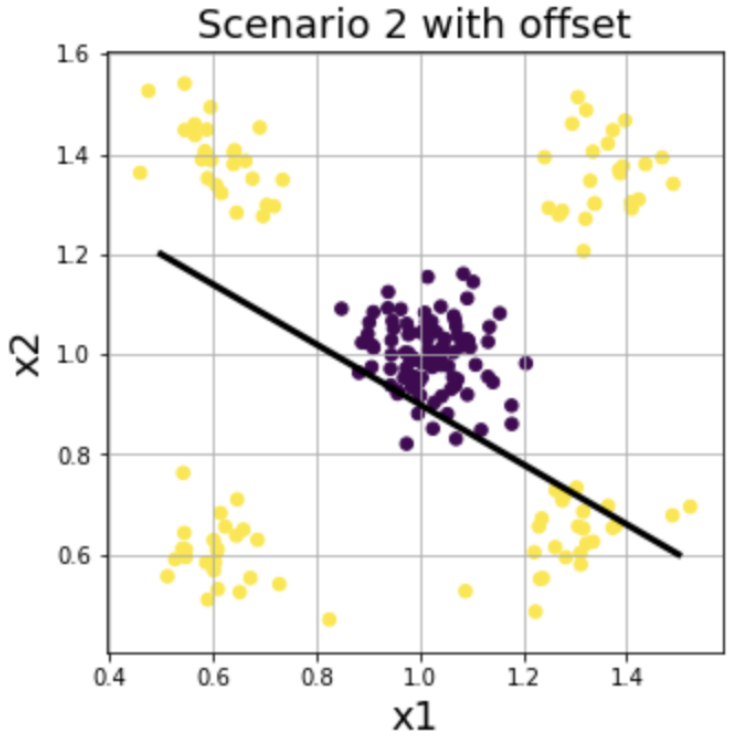
$r_{thresh} = \sqrt{0.1} = 0.31$ is a threshold which could separate the two classes.

Also $z_3 = x_2^2$ could do as well (or the absolute value)

Higher-Dimensional Feature Spaces

Exercise 1: Adding features

Same data with offset ?



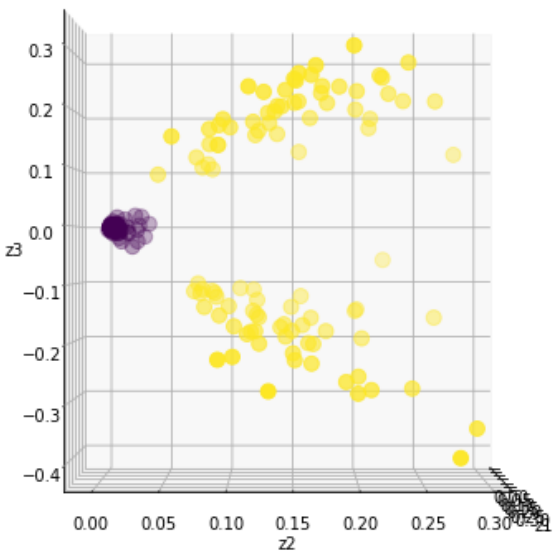
Use as feature: z_3 the distance to the point $x_c = [1, 1]$ (purple class)
$(z_3 = (x_1 - 1)^2 + (x_2 - 1)^2)$

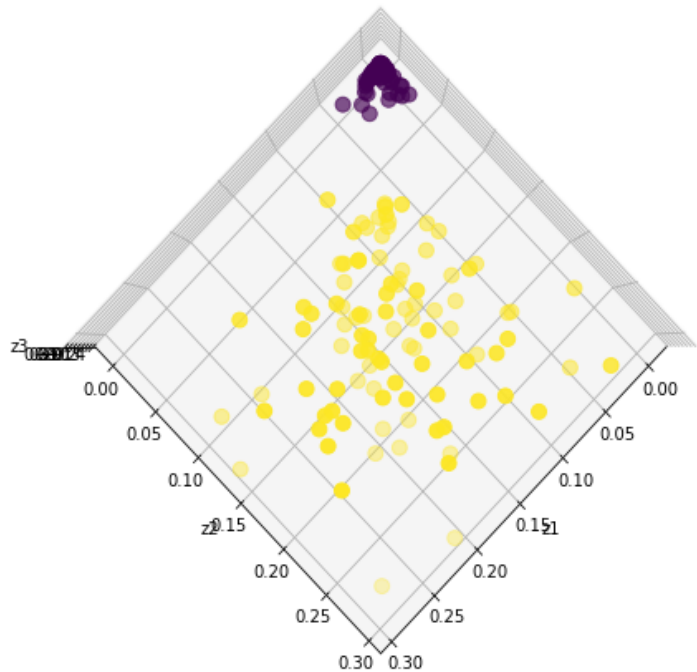
Kernels

Exercise 3: Kernel trick - Polynomial kernel

$\phi_{poly}(x) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]$

Applied to synthetic scenario 2:





Kernels trick

Exercice 5 : comparison

$$k(x_a, x_b) = (x_a \cdot x_b^T + c)^p \text{ and } k(x_a, x_b) = (x_a \cdot x_b^T)2 = \phi_{poly}(x_a).\phi_{poly}(x_b)$$

Q2: Thinking about the computational cost, and the generalization to higher order polynomials, explain which implementation is more recommended and why.

Answer

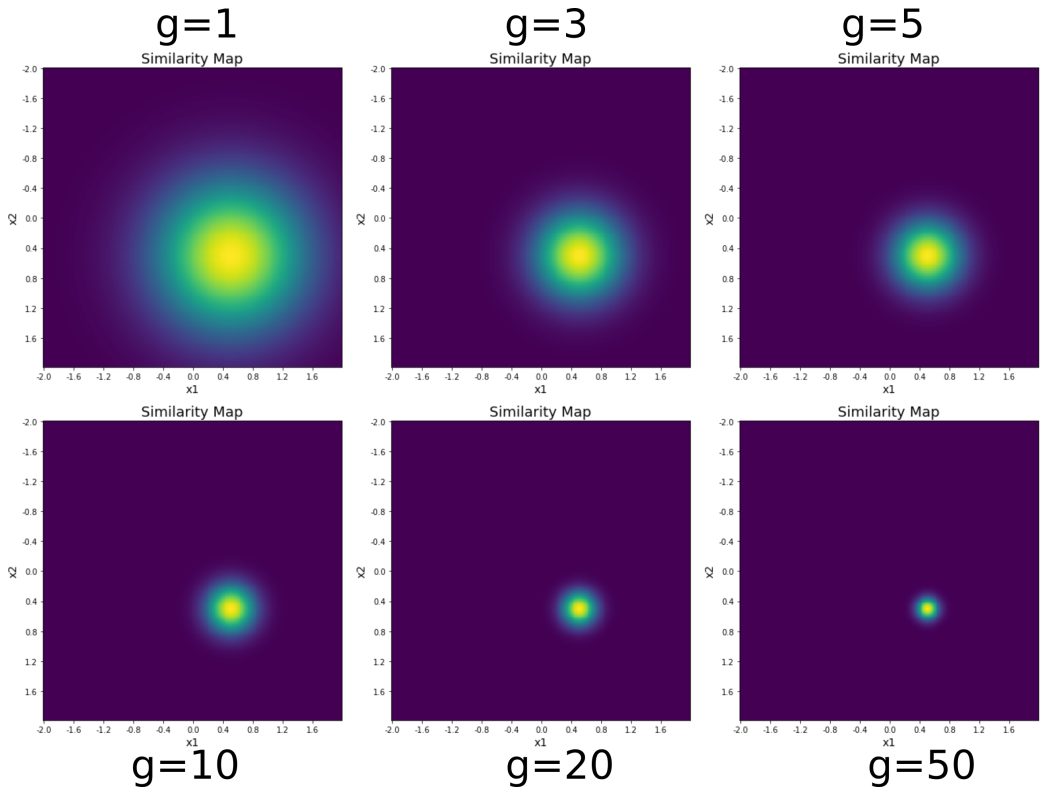
- when the order of the polynomial is high, the explicit projection needs to generate many features, before doing the dot product. In contrast, in the kernel case, the dot product is conducted in low dimension, and only one exponential is needed, so the kernal case will require less computations.
- if exploited in SVMs, the explicit projection might be better, as it will allow to **explicitly define the classifier as a linear classifier (i.e. with weights, which have some potential interpretation)**, so that computing the score for a new point will involve: projection + applying the linear classifier. Otherwise, the new score will be computed as a sum over the support vectors of kernel products, and the number of support vectors can be quite high.
- Note: in any case, both methods provide the same results.

Kernels

RBF similarity map

$k(x_a, x_b) = exp(-\gamma \|x_a - x_b\|^2)$

In the above, $\gamma > 0$ corresponds to the term $1/2\sigma^2$ of the typical Gaussian function. It is thus inversely proportional to the variance.

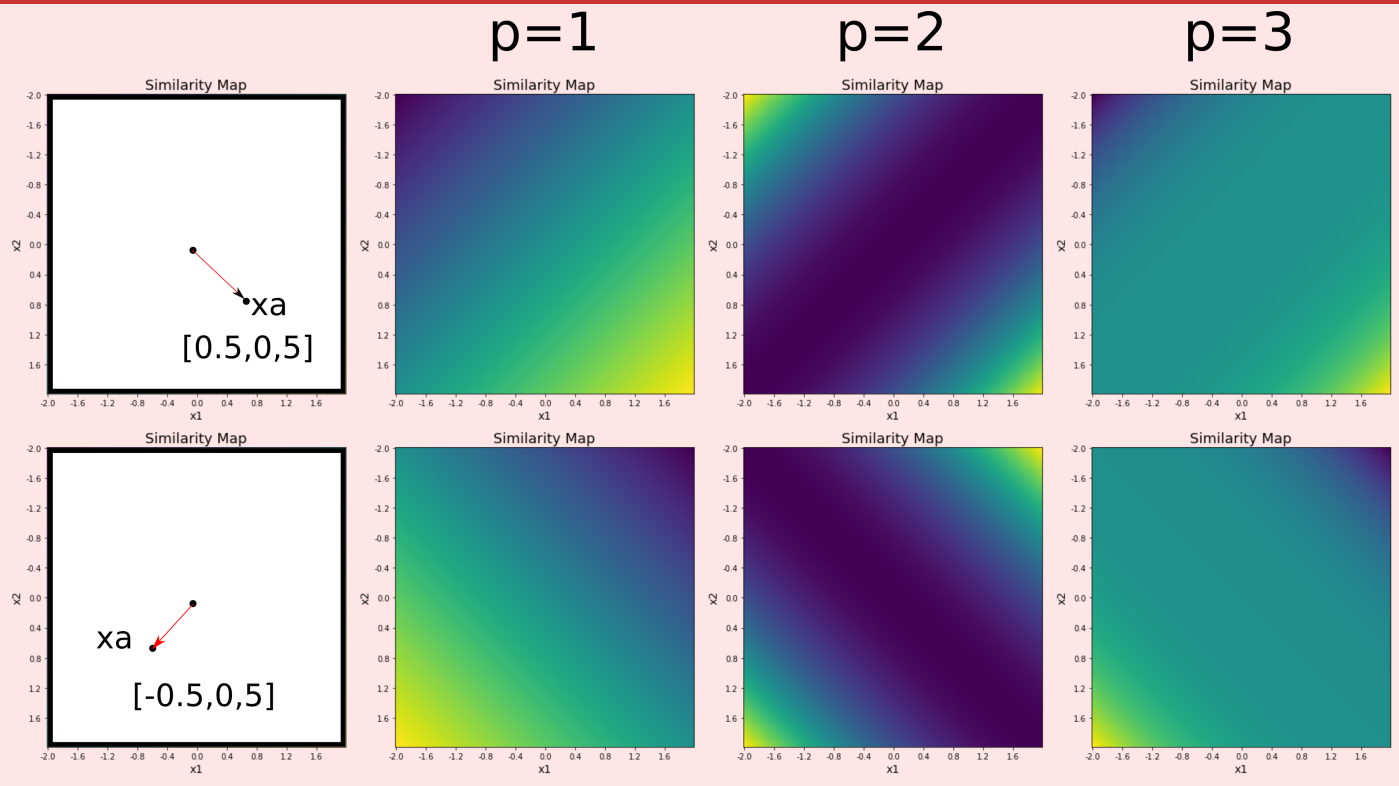


Impact of γ: from a classification point of view (e.g. with an SVM), when γ is high, this means that a datapoint with a given label will influence the classification of only nearby points, and not much points which are further away.

Kernels

Polynomial kernel similarity map: $k(x_a, x_b) = (x_a \cdot x_b^T + c)^p$

Solution: p=1,2,3



Similar points are note necessarily "nearby" points. Rather, what is captured is **a similarity in the direction**. Points which are in the same direction (or in the opposite direction as well for polynomial kernels of order 2,4,...) are more similar.

Note: this is similar with linear classifiers: points on line parallel to the decision line receive the same score

Comments

Comment

- Question 2 of exercise 7 is difficult.
- The last question of the lab is a bit unclear for me. I'm still not sure I really understood the question.
- The kernel map usage is not really clear. What use is there to know when the values are higher or lower? so this part of the lab is still not clear

Answer

The point of this exercise is to get an intuitive understanding of the kernel as a measure of similarity between vectors. But how do we define similarity? Is it the distance between vectors? The angle between them? Or something else?

We choose the appropriate kernel for our task based on how we expect to measure the similarity between vectors (considering what these vectors represent).

Comments

Comment

I would like to know how the function Phi (ϕ) which transforms the data in the new space is computed. For example, in exercice 1, how do we find phi function?

Answer

Well, we may decide on Phi (ϕ)

- by intuition
- in a systematic fashion (cf polynomial kernels: features = all potential products $\prod x_1^{k_1} ... x_n^{k_n}$ with $\sum k_i = p$)
- even in random ways: cf. random projection classifiers.
- but in our context: it is not defined - what is defined instead is the kernel (that we may select according to our need)

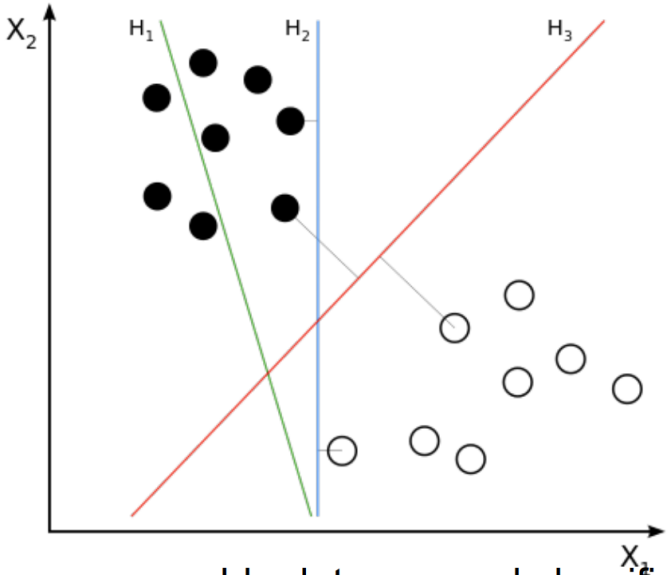
Support Vector Machines (SVM) - Part 1

SVM

Part 1

Some elements about SVMs

- notion of margin - find the linear classifier which has the largest margin

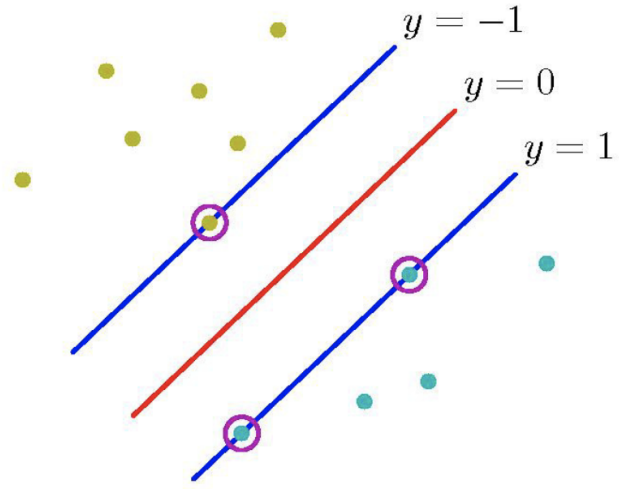


Main idea: look at the margin !

- H1: does not separate the classes
- H2: separate classes, but by a small margin
- H3: maximum margin

SVM

Part 1



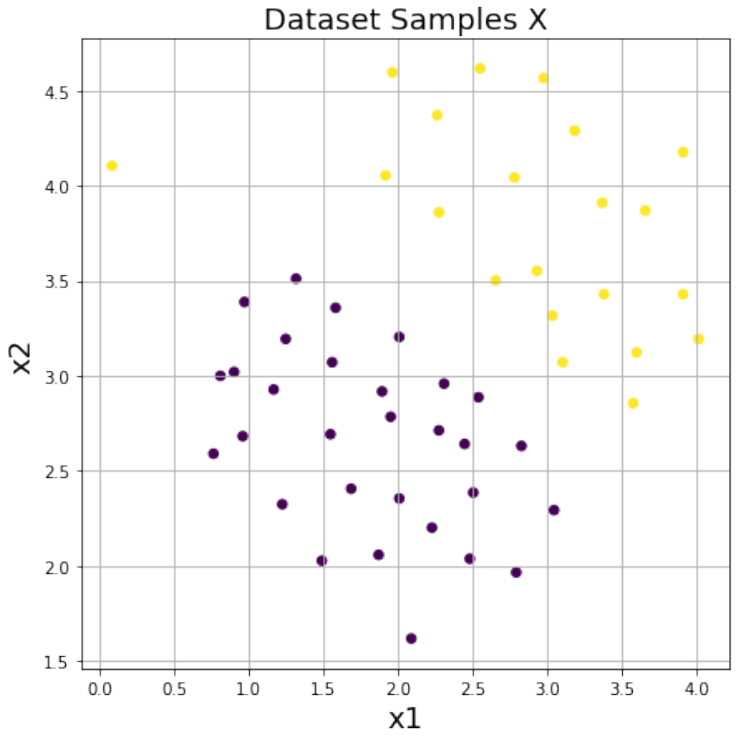
$$y(\mathbf{x}) = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + b = \sum_{i=1}^{N} a_i t_i \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}) + b = \sum_{i \in \mathcal{S}} a_i t_i k(\mathbf{x}_i, \mathbf{x}) + b$$

- sum over support vectors
- t_i - label of the support vector
- a_i overall importance of support vector in the decision
- $k(x_i, x)$ 'similarity' - how much support vector x_i influences the decision at point x
- kernel: implicit projection in a high dimensional space of the data point (cf figure)

Linear SVM

Exercise 1: Loading and visualize data

Do you think that the classes are linearly separable?



Solution

Yes, they are linearly separable, but whether we want that outlier to be classified correctly at the expense of the decision surface is another question.

Linear SVM

Exercise 3: Plotting the decision boundary

Recall that when using a linear SVM, the decision boundary is a line, the parameters of which can be derived from the optimized SVM model.

Q1: Provide the expression of the weights and bias of the boundary decision line as a function of the support vectors, their class, and the weights (a_i) learned during the SVM optimization. Provide the expression for the margin as well.

Solution (course notations):

The general expression is given by (cf. course):

$$y(x) = \sum_{i \in S} a_i t_i k(x_i, x) + b$$

Since in this case we use the linear kernel $k(x_i, x) = x_i \cdot x$, we have:

$$y(x) = \sum_{i \in S} a_i t_i x_i \cdot x + b = \left(\sum_{i \in S} a_i t_i x_i \right) \cdot x + b = w \cdot x + b$$

with
- $w = \sum_{i \in S} a_i t_i x_i$ - (and $b = \frac{1}{Ns} \sum_{i \in S} (t_i - \sum_{l \in S} a_l t_l x_l \cdot x_i) = \frac{1}{Ns} \sum_{i \in S} (t_i - w \cdot x_i)$)

The margin is given by: margin $= \frac{1}{\|w\|}$

Exercise 3: Plotting the decision boundary

Q1: Then compute the weights using this expression, and verify that they are the same than the coefficients provided by the SVM class (svm.coef_).

Solution (using scikit-learn output):

Using scikit-learn, the components of the attribute *dual_coef[0,:]* already contain the label (i.e. $a_i \times t_i$), so to obtain the weight vector w can be computed as:

$$w = \sum_{i \in S} \text{dual\_coef[0,i]} \quad x_i$$

This expression is computed in vectorial form in the next slide.

Exercise 3: Plotting the decision boundary

Q2: Then compute the weights using this expression, and verify that they are the same than the coefficients provided by the SVM class (svm.coef_).

Solution:

```
s = svm.support_vectors_ # Support vectors.
w = svm.dual_coef_[0,:] # The weights for the support vectors
wf = svm.coef_ # Coefficients of linear regression.
b = svm.intercept_ # Bias.

print(Regression coefficient)
print(wf)

w=np.reshape(w,(1,s.shape[0]))
wformula = np.dot(w,s)

print(Regression coefficient computed from formula: )
print(wformula)
```

Regression coefficient [[1.40718563 2.13398052]]

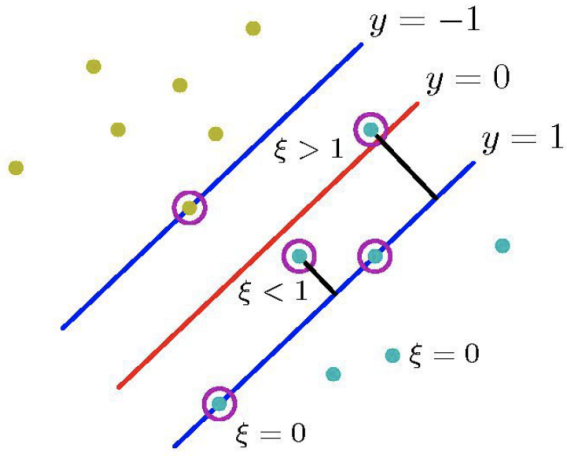
Regression coefficient computed from formula: [[1.40718563 2.13398052]]

Linear SVM -Influence of parameter C

- Primal problem

$$\begin{cases} \arg\min_{\mathbf{w},b} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i \right) & \text{subject to} \\ t_i y(\mathbf{x}_i) \geq 1 - \xi_i \qquad \forall i = 1, \ldots, N \\ \xi_i \geq 0 \end{cases}$$

$$\arg\min_{\mathbf{w},b} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \max(0, 1 - t_i y(\mathbf{x}_i)) \right)$$



- First term: measure the inverse of the margin
- Second term: count the margin violation
  - $\xi_i = 0$: point $\mathbf{x}_i$ well classified on good side
  - $0 < \xi_i < 1$: point well classified, but within the margin
  - $\xi_i > 1$: point badly classified

Note:

- Optimization: compromise between the two
- C: when it is very very large, we don't tolerate margin violation/errors $\Rightarrow$ recover the separable case (when possible)

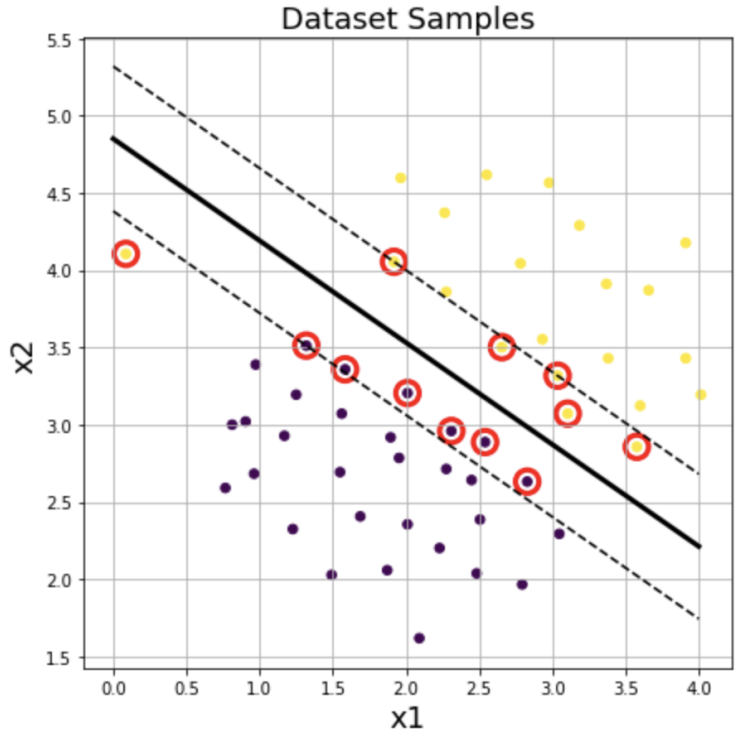
Linear SVM

Exercise 4: Influence of Parameter C

Run the code to learn the linear SVM classifier on the 2D dataset samples (X). In particular, experiment with different values of the cost parameter $C \in \{1, 10, 20, 100, 200, 300\}$. As C increases, see the evolution of the margin, the number of support vectors, the accuracy.

Solution: C=1

```
Classification accuracy: 0.980
Num. support vectors: 12
Margin = 0.391
```



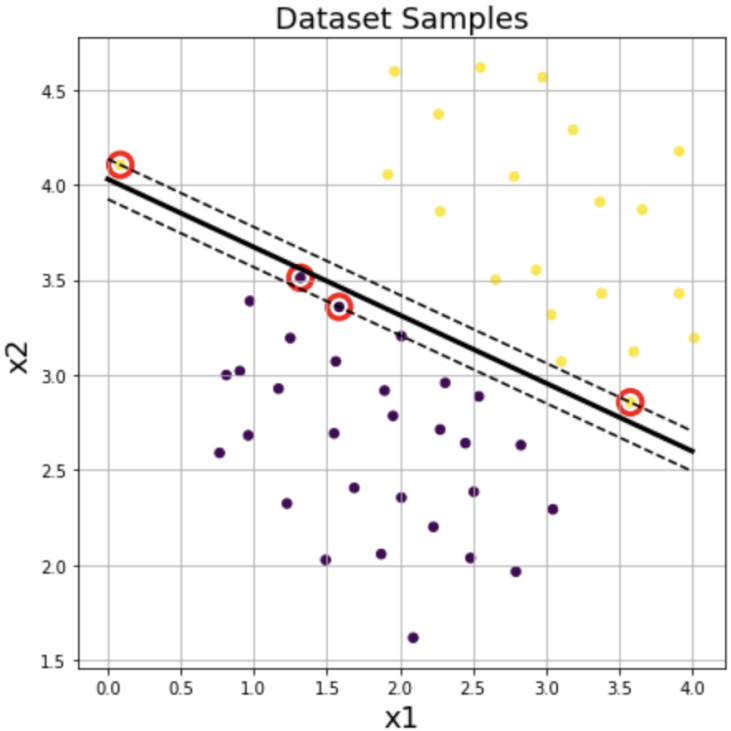
Linear SVM

Exercise 4: Influence of Parameter C

Run the code to learn the linear SVM classifier on the 2D dataset samples (X). In particular, experiment with different values of the cost parameter $C \in \{1, 10, 20, 100, 200, 300\}$. As C increases, see the evolution of the margin, the number of support vectors, the accuracy.

Solution: C=50

```
Classification accuracy: 1.000
Num. support vectors: 4
Margin = 0.100
```



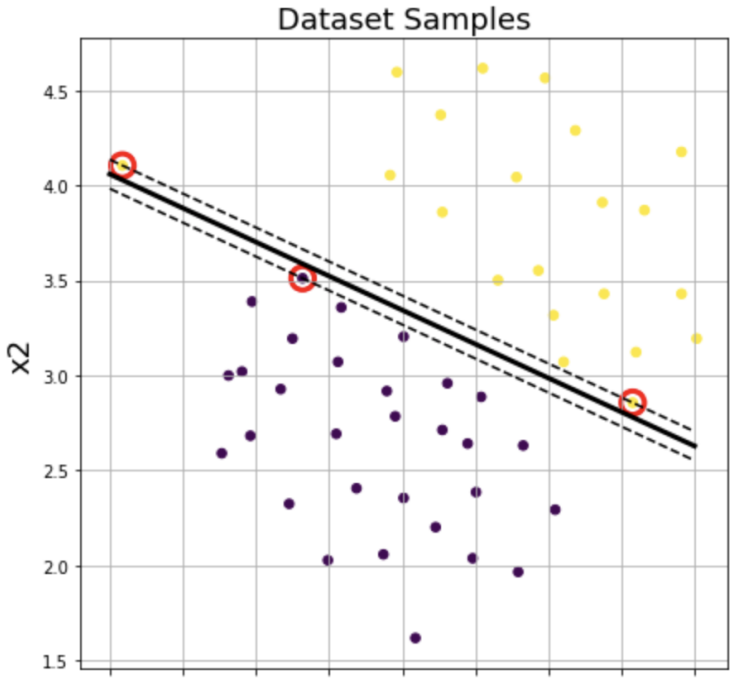
Linear SVM

Exercise 4: Influence of Parameter C

Run the code to learn the linear SVM classifier on the 2D dataset samples (X). In particular, experiment with different values of the cost parameter $C \in \{1, 10, 20, 100, 200, 300\}$. As C increases, see the evolution of the margin, the number of support vectors, the accuracy.

Solution: C=100

```
Classification accuracy: 1.000
Num. support vectors: 3
Margin = 0.072
```



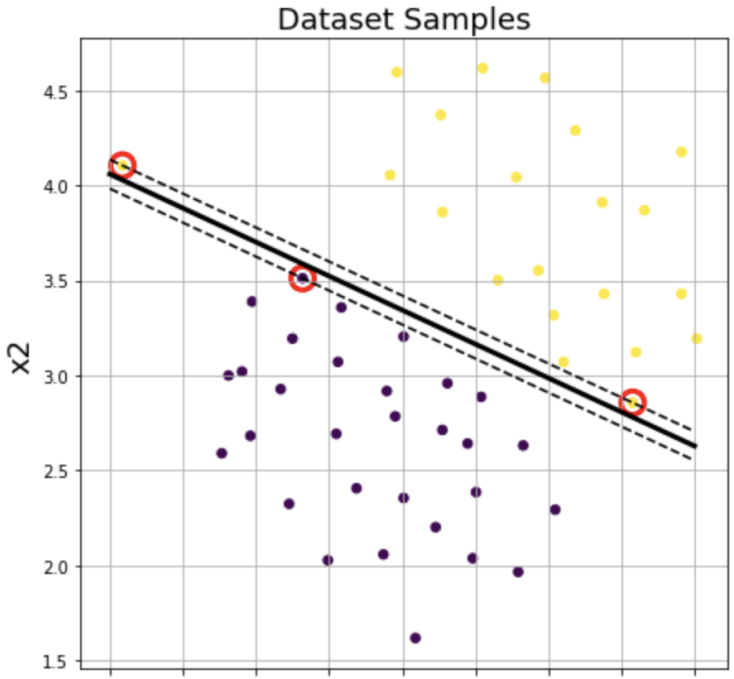
Linear SVM

Exercise 4: Influence of Parameter C

Run the code to learn the linear SVM classifier on the 2D dataset samples (X). In particular, experiment with different values of the cost parameter $C \in \{1, 10, 20, 100, 200, 300\}$. As C increases, see the evolution of the margin, the number of support vectors, the accuracy.

Solution: C=300

```
Classification accuracy: 1.000
Num. support vectors: 3
Margin = 0.072
```



Linear SVM

Exercise 4: Parameter C

Q2: Which difference do you observe in the learned hyperplanes and other elements (margin, number of support vectors) for different values of C ? Given the interpretation of C is this evolution normal? Explain.

$$\arg\min_{\mathbf{w},b}\left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i\right)$$

Answer

As C goes from 1 to 300, we observe that:

- the accuracy increases (the number of errors tends to decrease; normal w.r.t. interpretation of C)
- the margin decreases (normal w.r.t. interpretation of C)
- the number of support vectors tends to decrease (ok with interpretation of C; indeed, more and more points fall on the margin or beyond)

Exercise 4: Parameter C

Q3: What do you observe for C values above 100? What can you say about the slack variables?

Answer

In that case, the accuracy is 100%, and the margin does not evolve. This means that all data points are well classified and on the margin or beyond. So the slack variables are all 0. Hence the solution is always exactly the same, as increasing C has no more impact on the function to optimize.

Note: how to interpret that the margin is different for $C = 50$ and $C = 100$ although the training accuracy is 100% in both cases?

For $C = 50$, there is a data point that is correctly classified but within the margin. This means that its slack variable is strictly positive (between 0 and 1). So even if the accuracy is already 100%, the decision function can still change if we increase C. This will cause the slack variable of this point to be 0, and thus decrease the margin further.

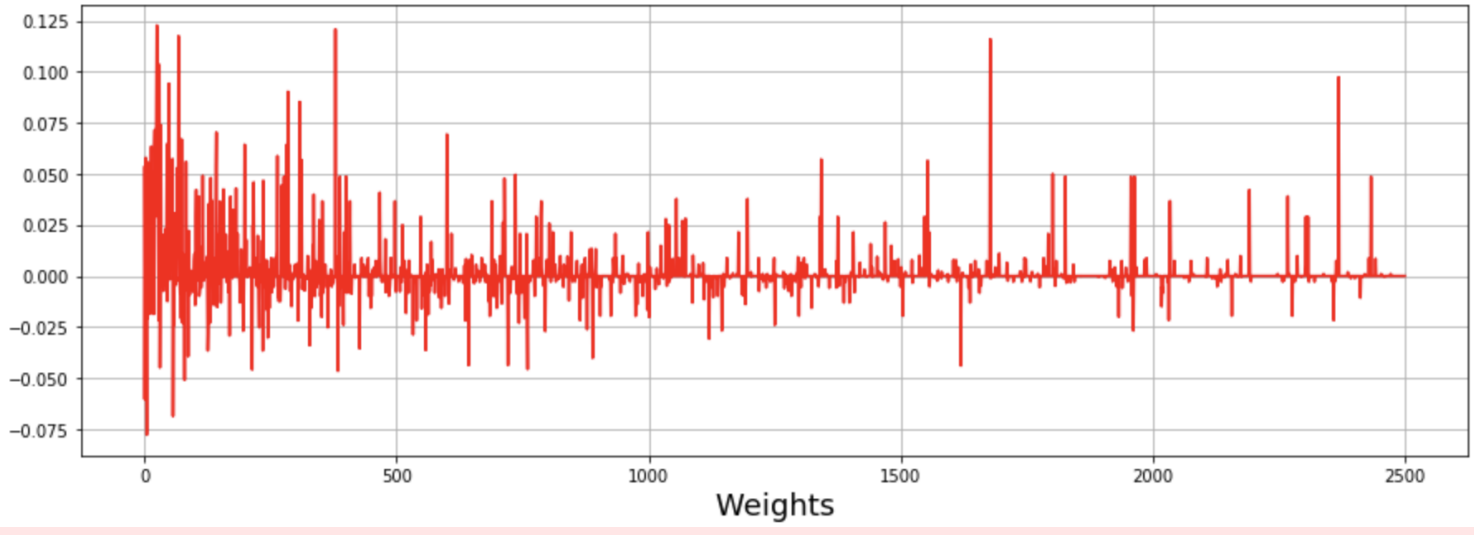
Linear SVM - Spam/Non-Spam

Exercise 5: Learn and test SVMs

Using the e-mail spam dataset, learn a linear SVM classifier on the training set and apply it on the test set. Repeat for different sizes of the training data: 50, 100, 400 and 702 samples.

Solution: 50 samples

```
Training with 50 samples
Email classification:
+ Num. train samples 50
+ Training classification accuracy: 1.000
+ Testing classification accuracy: 0.796
+ Num. support vectors: 33
```



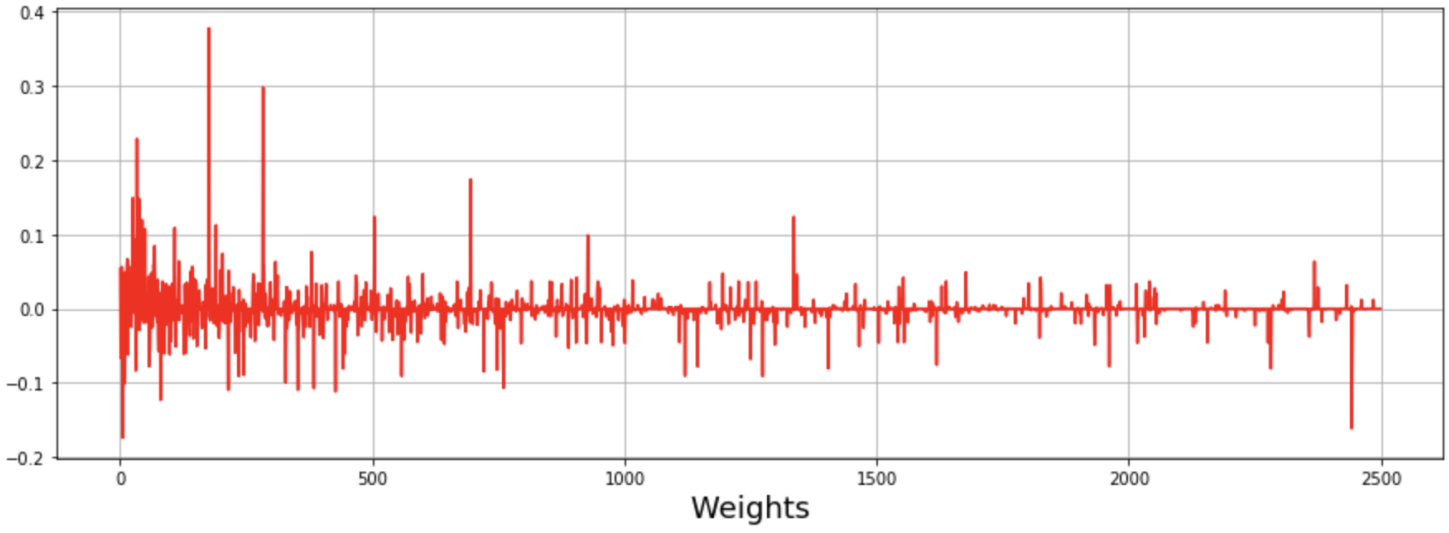
Linear SVM - Spam/Non-Spam

Exercise 5: Learn and test SVMs

Using the e-mail spam dataset, learn a linear SVM classifier on the training set and apply it on the test set. Repeat for different sizes of the training data: 50, 100, 400 and 702 samples.

Solution: 100 samples

```
Training with 100 samples
Email classification:
+ Num. train samples 100
+ Training classification accuracy: 1.000
+ Testing classification accuracy: 0.962
+ Num. support vectors: 51
```



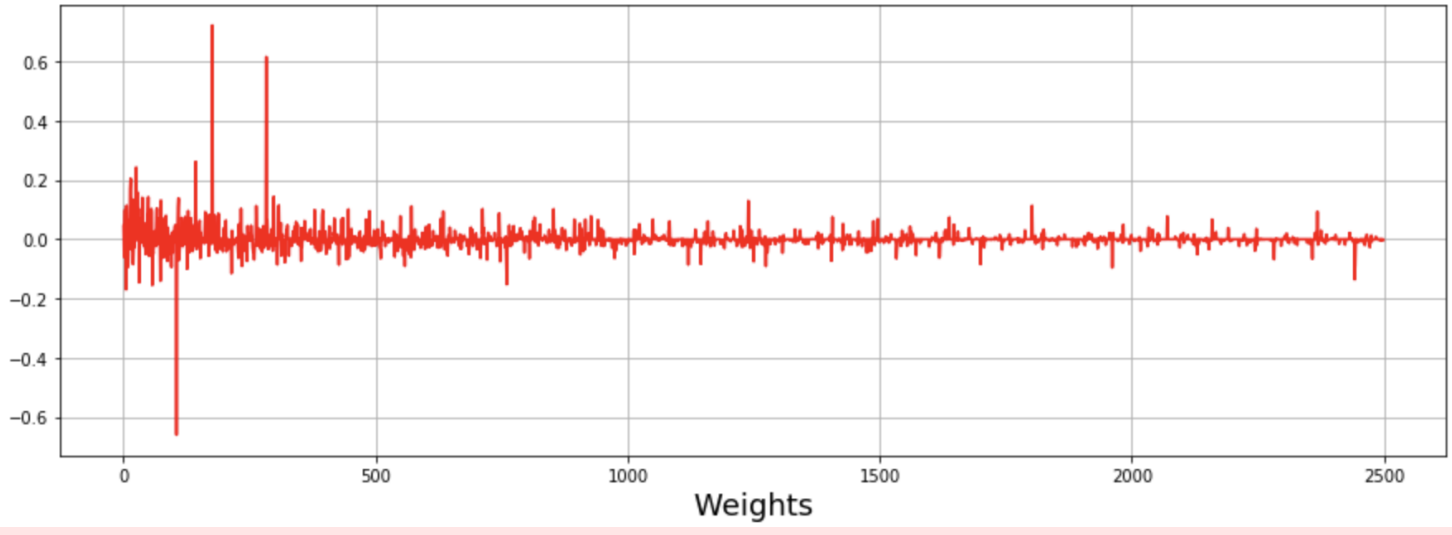
Linear SVM - Spam/Non-Spam

Exercise 5: Learn and test SVMs

Using the e-mail spam dataset, learn a linear SVM classifier on the training set and apply it on the test set. Repeat for different sizes of the training data: 50, 100, 400 and 702 samples.

Solution: 400 samples

```
Training with 400 samples
Email classification:
+ Num. train samples 400
+ Training classification accuracy: 1.000
+ Testing classification accuracy: 0.931
+ Num. support vectors: 104
```



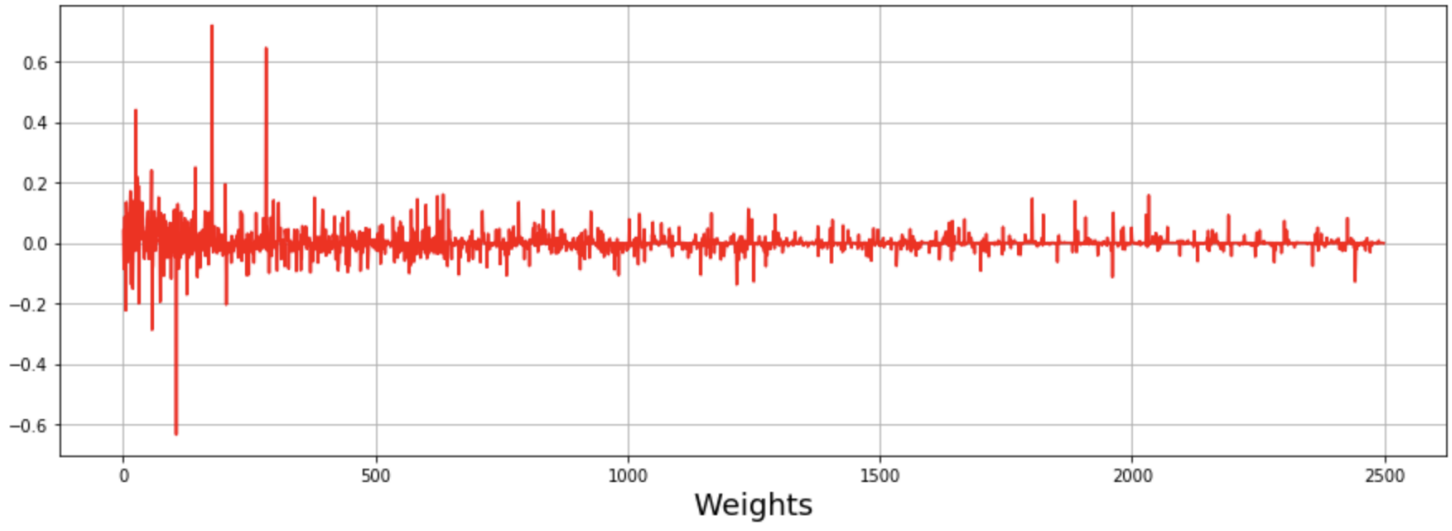
Linear SVM - Spam/Non-Spam

Exercise 5: Learn and test SVMs

Using the e-mail spam dataset, learn a linear SVM classifier on the training set and apply it on the test set. Repeat for different sizes of the training data: 50, 100, 400 and 702 samples.

Solution: 702 samples

```
Training with 702 samples
Email classification:
+ Num. train samples 702
+ Training classification accuracy: 1.000
+ Testing classification accuracy: 0.950
+ Num. support vectors: 145
```



Linear SVM

Exercise 5: Learn and test SVMs

Q2: Describe how the classification accuracies (train and test) evolve with the number of training samples. Explain what you observe with the training accuracy, and the test accuracy.

Answer

- the training accuracy is always 1, showing that in this high-dimensional space (2500), it always possible to obtain a perfect training accuracy (when the number of samples is lower than the number of dimensions).
- the test accuracy reaches 0.95/0.96, showing that there exist good linear classifier to separate the data.
- the testing accuracy is relatively small when working with 50 documents only, but quickly reaches a good score as well, demonstrating the capacity of the SVM (and of the margin concept) to provide good solutions.
- accordingly, the number of support vector increases

Linear SVM

Exercise 5: Learn and test SVMs

Q3: How do the learned weight evolve? Do you observe any specific pattern as the number document is changing?

Answer

- the weights are more variable across features for N=50 samples.
- for larger N, the weights are larger for features with a lower index, and we do observe that for some index, the weight remain more or less the same, showing their importance for spam/non-spam classification

Exercise 5: Learn and test SVMs

Q4: What is the proportion of support vectors compared to the number of training samples, as N increases? Are these sparse solutions (i.e. usually, we would assume that sparse is achieved when less than 10% of the samples are used)? Can you explain why it is like this?

Answer

These solutions are not that sparse, essentially due to the number of training samples versus the feature dimension. Nevertheless, we observe that the sparsity is improving (20% for $N = 700$, vs 66% for $N = 50$).

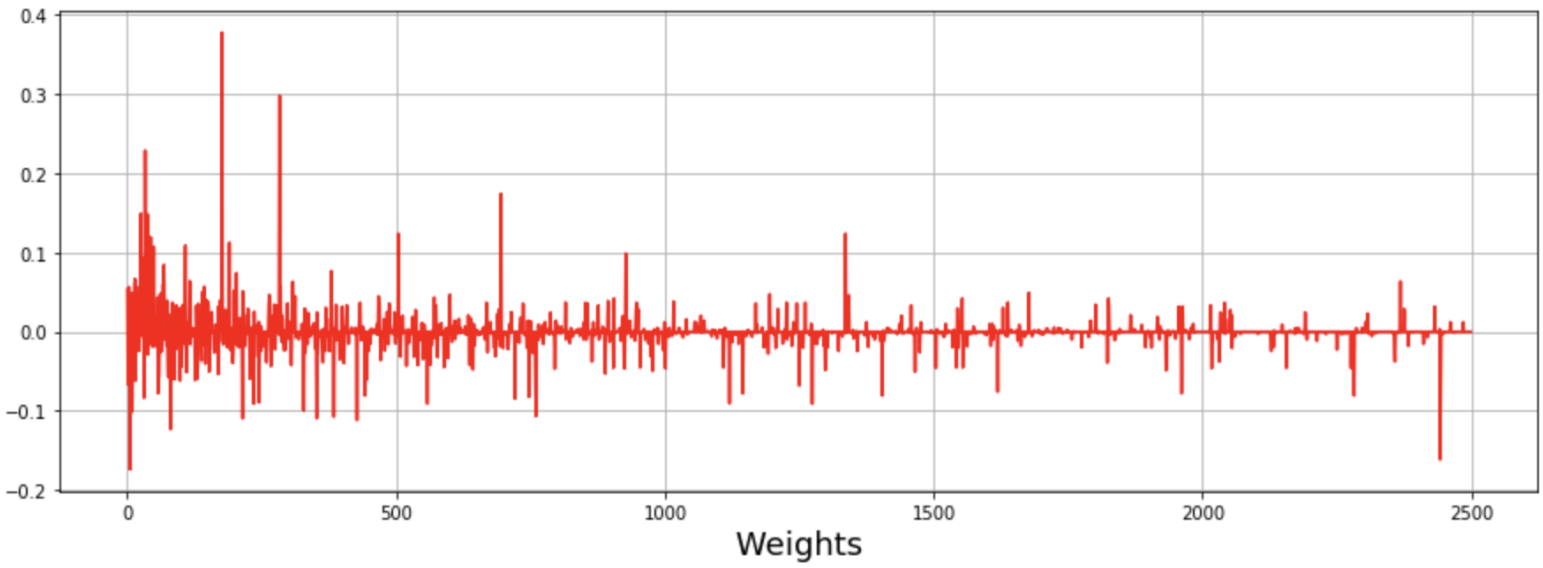
Linear SVM

Exercise 5: Learn and test SVMs

Q5: For the same set of 100 documents, report the training and classification accuracy for C=1, 10 or 100. How do the test accuracy and learned weights evolve? What can you notice? Explain the results.

Solution: for C=1,10,100, we obtain:

```
Training with 100 samples
Email classification:
+ Num. train samples 100
+ Training classification accuracy: 1.000
+ Testing classification accuracy: 0.962
+ Num. support vectors: 51
```



The results are the same for all C values. Indeed, for the same set of samples, there are no errors (accuracy of 1), and no samples within the margin. Thus the optimal margin has been found, and nothing will change.

Linear SVM

Poll - Which of the following options are true?

- 1) The SVM is not able to classify properly both classes.
- 2) When C is large there is no tolerance for errors, so it is automatic that the classification is perfect on the training data.
- 3) The fact that the solutions are the same for different values of C is only true because the initialization of the SVM optimization algorithm is the same.
- 4) The fact that when C=1 the classification is already perfect on the training data indicate that we have found the optimal margin (the error term linked to C is 0). So, further increasing C will not further affect the solution (and decision boundary).

Linear SVM

SVM on Spam/Non-Spam

- 1) The SVM is not able to classify properly both classes.
- 2) When C is large there is no tolerance for errors, so it is automatic that the classification is perfect on the training data.
- 3) The fact that the solutions are the same for different values of C is only true because the initialization of the SVM optimization algorithm is the same.
- 4) The fact that when C=1 the classification is alread perfect on the training data indicate that we have found the optimal margin (the error term linked to C is 0). So, further increasing C will not further affect the solution (and decision boundary).

Solution

1) True, since the test results are not perfect (but this depends on how we feel about 'properly').

Linear SVM

SVM on Spam/Non-Spam

- 1) The SVM is not able to classify properly both classes.
- 2) When C is large there is no tolerance for errors, so it is automatic that the classification is perfect on the training data.
- 3) The fact that the solutions are the same for different values of C is only true because the initialization of the SVM optimization algorithm is the same.
- 4) The fact that when C=1 the classification is alread perfect on the training data indicate that we have found the optimal margin (the error term linked to C is 0). So, further increasing C will not further affect the solution (and decision boundary).

Solution

1) True, since the test results are not perfect (but this depends on how we feel about 'properly').

2) False, as there is no guarantee in general that we can obtain a perfect training score, even for very large C.

Linear SVM

SVM on Spam/Non-Spam

- 1) The SVM is not able to classify properly both classes.
- 2) When C is large there is no tolerance for errors, so it is automatic that the classification is perfect on the training data.
- 3) The fact that the solutions are the same for different values of C is only true because the initialization of the SVM optimization algorithm is the same.
- 4) The fact that when C=1 the classification is alread perfect on the training data indicate that we have found the optimal margin (the error term linked to C is 0). So, further increasing C will not further affect the solution (and decision boundary).

Solution

1) True, since the test results are not perfect (but this depends on how we feel about 'properly').

2) False, as there is no guarantee in general that we can obtain a perfect training score, even for very large C.

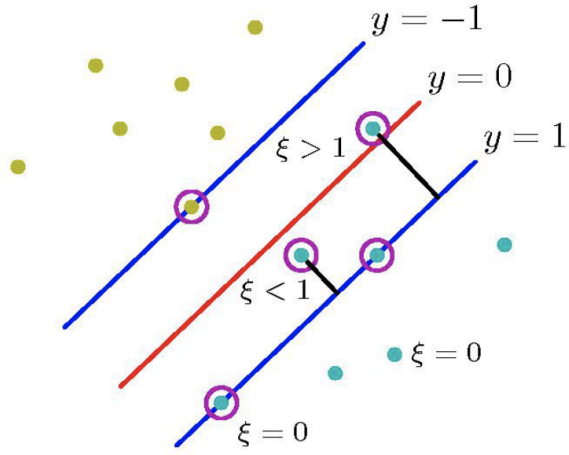
3) False, as the SVM solution is obtained by solving a quadratic programming problem which has a unique solution, so it does not depend on any initilization.

Linear SVM

- Primal problem

$$\begin{cases} \arg\min_{\mathbf{w},b} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i \right) & \text{subject to} \\ t_i y(\mathbf{x}_i) \geq 1 - \xi_i \qquad \forall i = 1, \ldots, N \\ \xi_i \geq 0 \end{cases}$$

$$\arg\min_{\mathbf{w},b} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \max(0, 1 - t_i y(\mathbf{x}_i)) \right)$$



Exercise 5: Learn and test SVMs

- 4) The fact that when C=1 the classification is alread perfect on the training data indicate that we have found the optimal margin (the error term linked to C is 0). So, further increasing C will not further affect the solution (and decision boundary).

Solution

4) False. The fact that the classification is perfect only indicates that $\xi_i < 1$ for all points. Hence the optimization of the margin may still be improved (in the sense of increasing it), by decreasing C (since decreasing C will reduce the cost or margin violations and thus indirectly put more weight on increasing the margin term in the overal optimization).

Since we do not observe any change in the margin, this actually implies that for $C = 1$, there are no errors (accuracy of 1), and no samples within the margin. Thus the optimal margin has been found, and nothing will change.

Linear SVM

Exercise 6: Interpretation

Q1: When using all documents for training: What can you say about the weights of the words? Does it make sense? Which words influence more the class "spam"? Which words influence more the class "non spam"?

Answer

- The words more related to 'selling' are appearing as an indicator of spam (call, click, website to visit, http, extension com, free) or adult content (adult, live ?)
- Those related to normal content (i.e. dataset collected on a university campus) like (thanks, edu extension, language, linguistic, linguist, university) are pushing toward the non-spam class

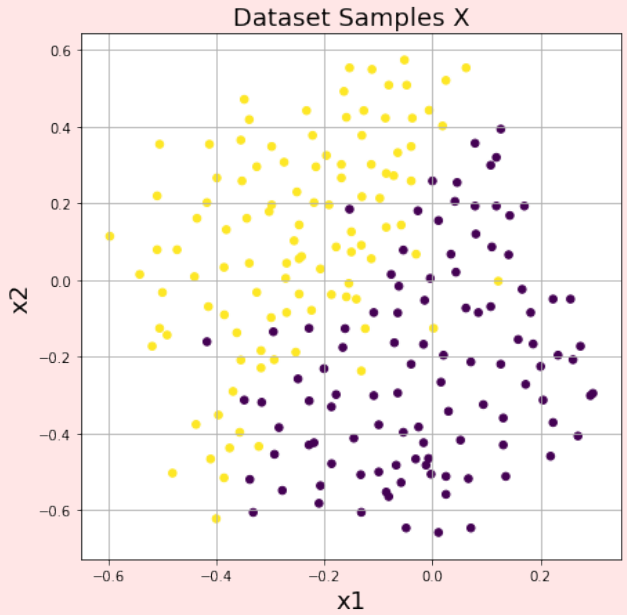
Support Vector Machines (SVM) - Part 2

SVM with RBF kernel

Exercise 2: 2D Dataset

A two-dimensional dataset is loaded and visualized.
Q1: Do you think that the classes are easily separable?

Solution: several answers possible



- no: there is apparently a linear separation, but many datapoints from both classes are very close to each other, so in any case, there will be classification errors, including the points which fall on the other side of the potential linear decision boundary (which can be due to the expected noise associated to each class, and thus those points are points drawn from the class distribution).
- yes, it it seems that there is a linear separation between the two classes, although some points will probably be misclassified at training time, if we select the right set of parameters. Otherwise, this may lead to overfitting.

SVM with RBF kernel

Exercise 3: Hyperparameters values

Try with $C = 1, 1000$ and $\gamma = 1, 10, 100, 1000$. Visualize the learned models with different combinations of hyperparameter values. Remember that the kernel indicates how close are two data points and whould be considered as 'neighboors' (and thus be labeled similarily).

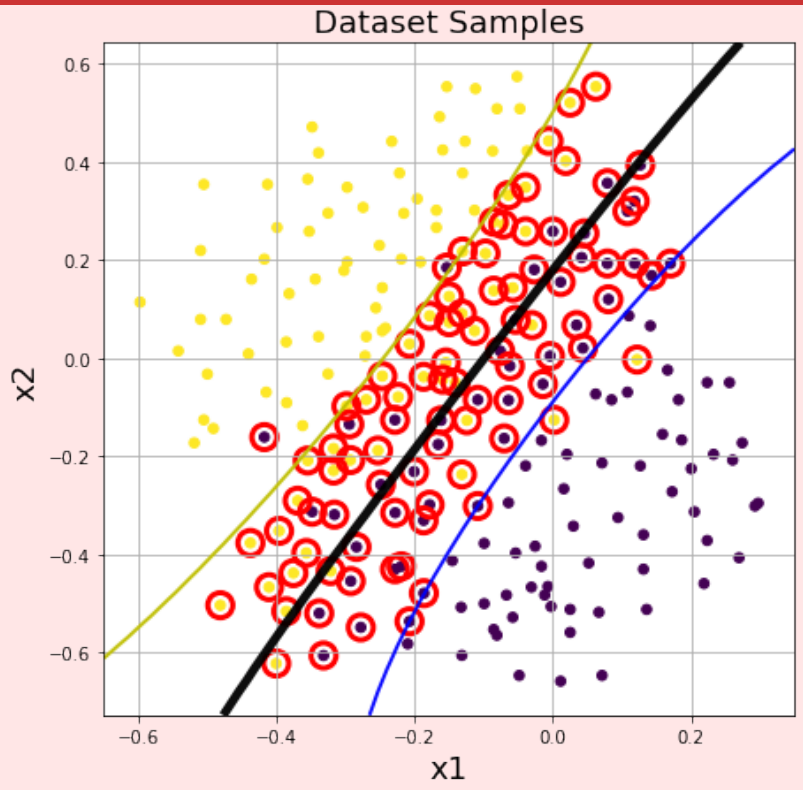
Have a look at the boundary decision functions, training accuracy, as well as at the number of support vectors in the different configurations.

Q1: Which set of parameters sounds problematic or good here (in particular think of underfitting and overfitting)? Explain why in each case (i.e. why the specific set of C and γ values have these effects).

SVM with RBF kernel

Exercise 3: Hyperparameters values

Solution: $C = 1$ and $\gamma = 1$

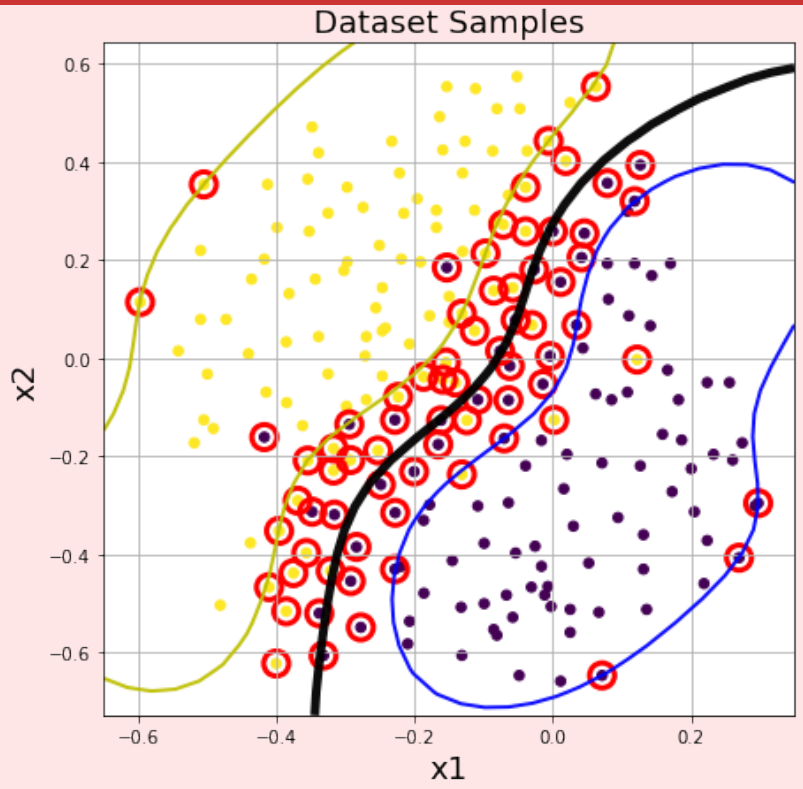


Training classification accuracy: 0.919
Num. support vectors: 94

SVM with RBF kernel

Exercise 3: Hyperparameters values

Solution: $C = 1$ **and** $\gamma = 10$

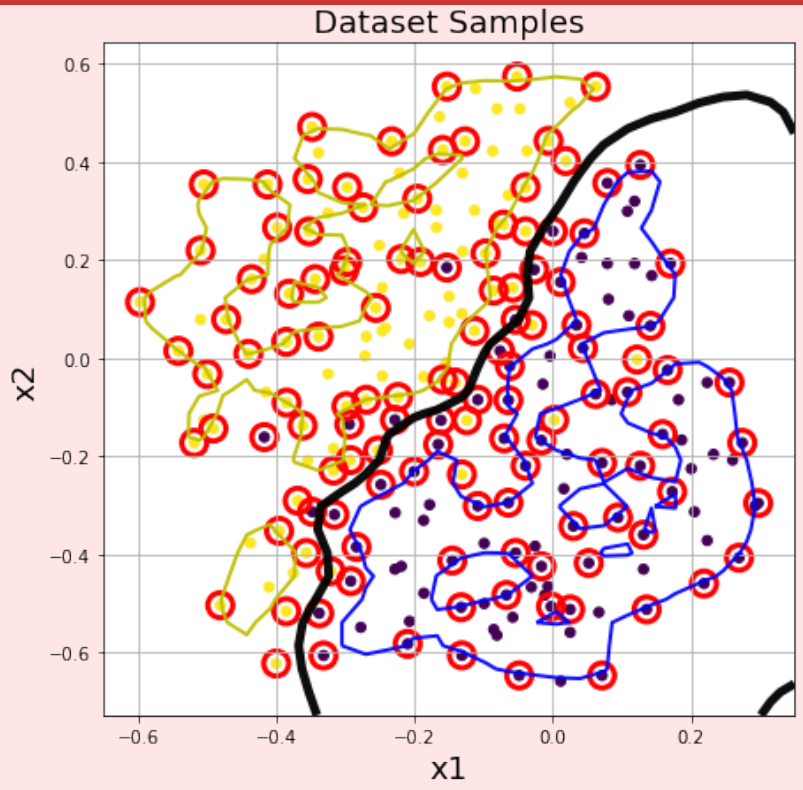


Training classification accuracy: 0.934
Num. support vectors: 73

SVM with RBF kernel

Exercise 3: Hyperparameters values

Solution: $C = 1$ **and** $\gamma = 100$

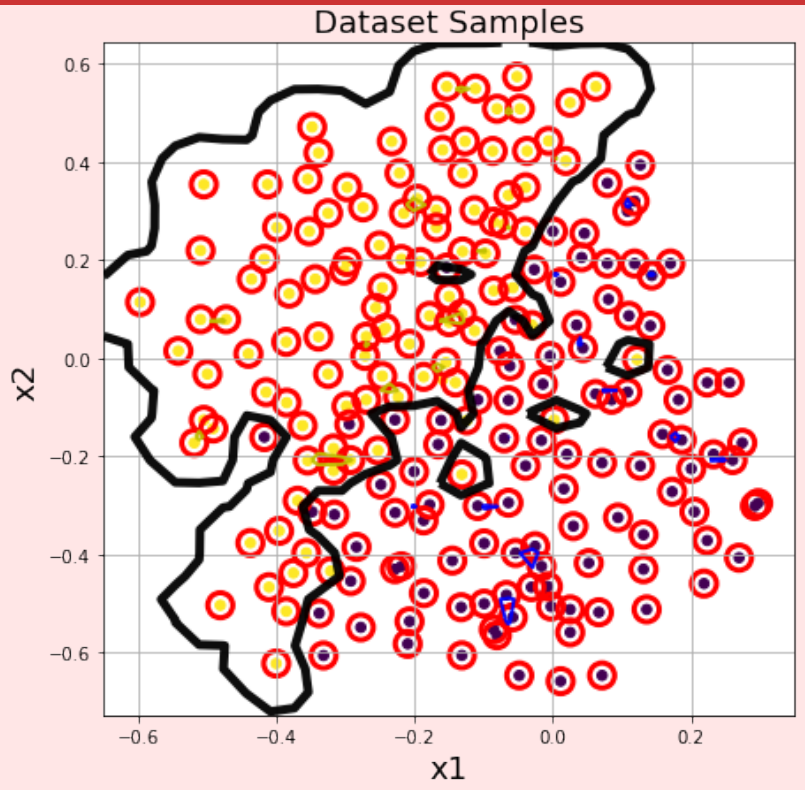


Training classification accuracy: 0.943
Num. support vectors: 126

SVM with RBF kernel

Exercise 3: Hyperparameters values

Solution: $C = 1$ **and** $\gamma = 1000$



Training classification accuracy: 1.0
Num. support vectors: 208

SVM with RBF kernel

Exercise 3: Hyperparameters values

Q1: Which set of parameters sounds problematic or good here (in particular think of underfitting and overfitting)? Explain why in each case (i.e. why the specific set of C and γ values have these effects).

Answer: $C = 1$

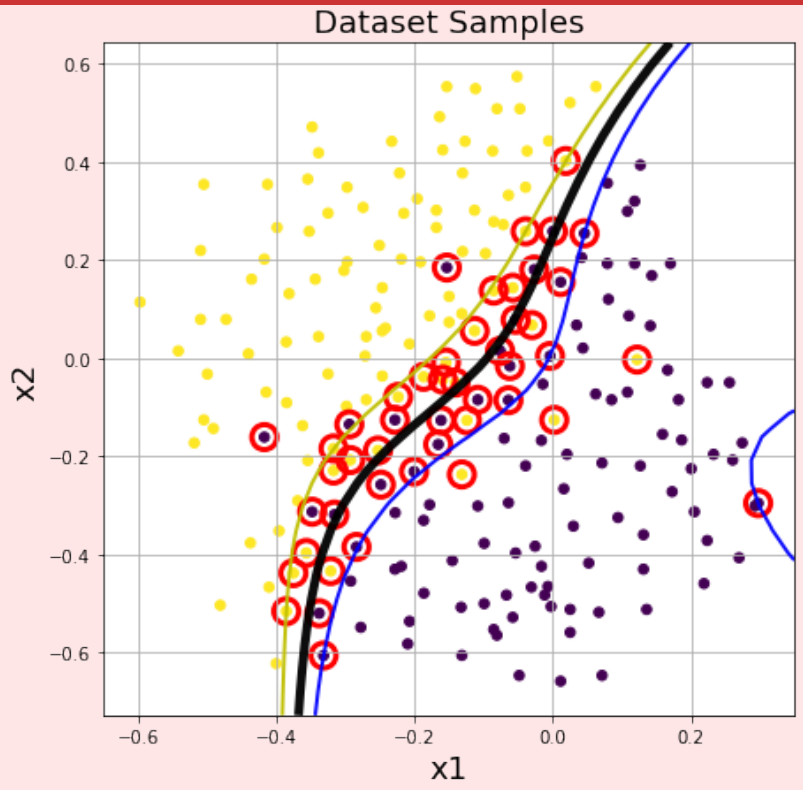
For $C = 1$, which does not penalize much the errors:

- number of support vectors: no systematic trend (values = 94, 73, 126, 208 as gamma increases). Nevertheless, as a ternd, when gamma is large, there are often more support vectors
- accuracy increases as gamma increase (0.919, 0.934, 0.943, 1). However, we observe underfitting (gamma=1) clear overfitting for gamma = 1000 (small overfitting for gamma=100).
- best: gamma=10: follows almost the linear trend

SVM with RBF kernel

Exercise 3: Hyperparameters values

Solution: $C = 1000$ **and** $\gamma = 1$

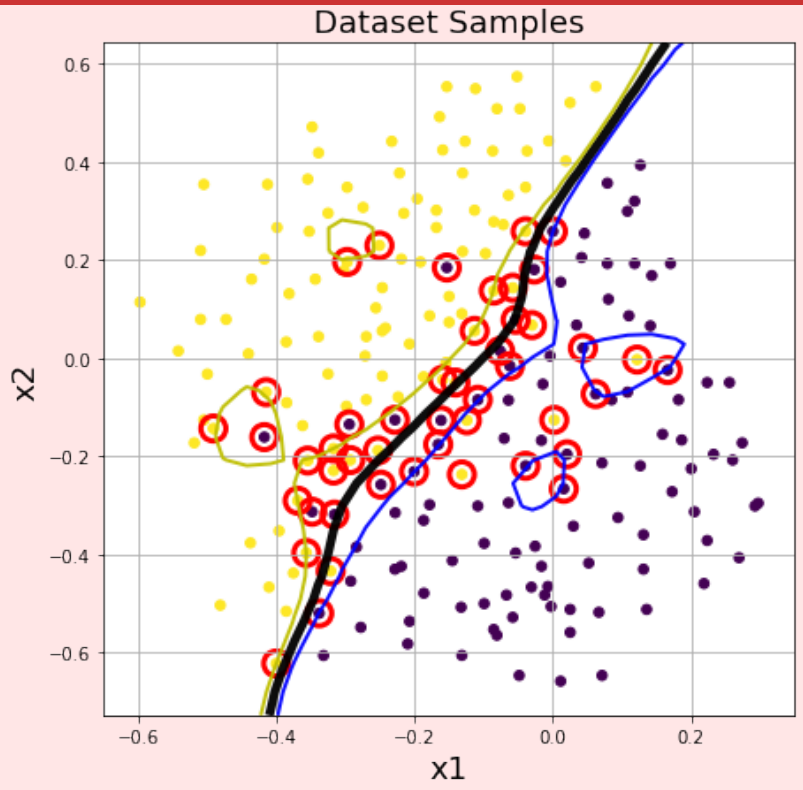


Training classification accuracy: 0.934
Num. support vectors: 47

SVM with RBF kernel

Exercise 3: Hyperparameters values

Solution: $C = 1000$ **and** $\gamma = 10$

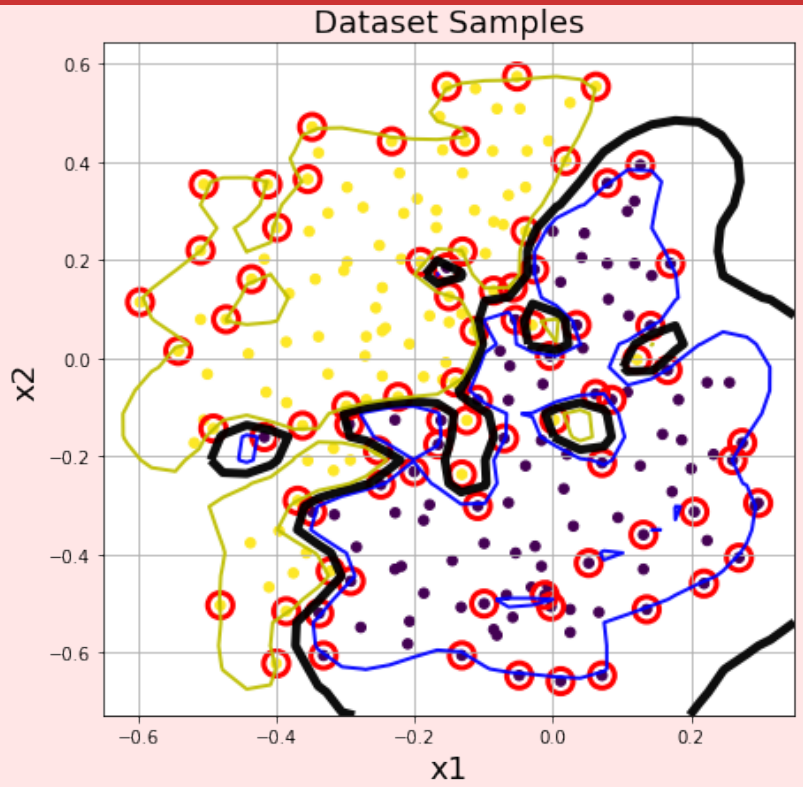


Train classification accuracy: 0.934
Num. support vectors: 47

SVM with RBF kernel

Exercise 3: Hyperparameters values

Solution: $C = 1000$ **and** $\gamma = 100$

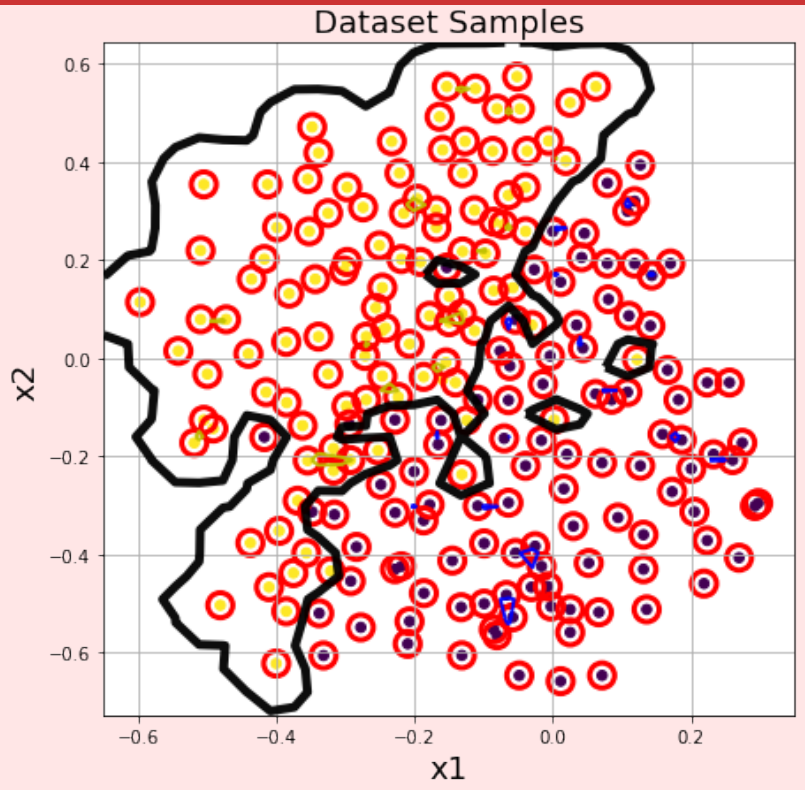


Training classification accuracy: 1.0
Num. support vectors: 81

SVM with RBF kernel

Exercise 3: Hyperparameters values

Solution: $C = 1000$ **and** $\gamma = 1000$



Train classification accuracy: 1.0
Num. support vectors: 208

SVM with RBF kernel

Exercise 3: Hyperparameters values

Q1: Which set of parameters sounds problematic or good here (in particular think of underfitting and overfitting)? Explain why in each case (i.e. why the specific set of C and γ values have these effects).

Answer: $C = 1000$

For $C = 1000$:

- for gamma =1 : good results overall: (47 support vectors, accuracy = 0.934). The decision function is quite regular and smooth, as it integrates information coming from further points (compared to C=1, gamma=10)
- gamma = 10: results are ok (47 support vectors, accuracy = 0.934), close to overfitting, although decision boudary does not change much compared to gamma=1
- gamma = 100. overfitting (81 support vectors, accuracy = 1) - compared to $C = 1$, we see thar more penalization of errors lead to higher training accuracy.
- gamma = 1000: overfitting, results vers similar to gamma=1000 and $C = 1$. Everypoint becomes a support vector (accuracy = 1, #support vectors = 208).

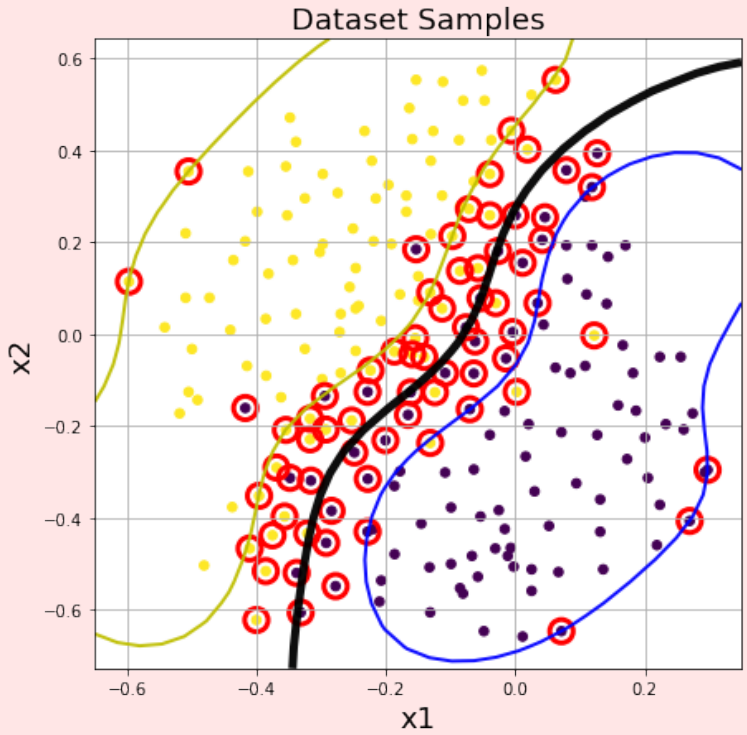
SVM with RBF kernel

Exercise 3: Hyperparameters values

Q1: Which set of parameters sounds good here?

Answer: $C = 1$ and $\gamma = 10$ looks ok.

The resulting decision boundary is smooth and it is a good compromise between large margin and some margin violations. It is more convenient for generalization.



Train classification accuracy: 0.934
Num. support vectors: 73

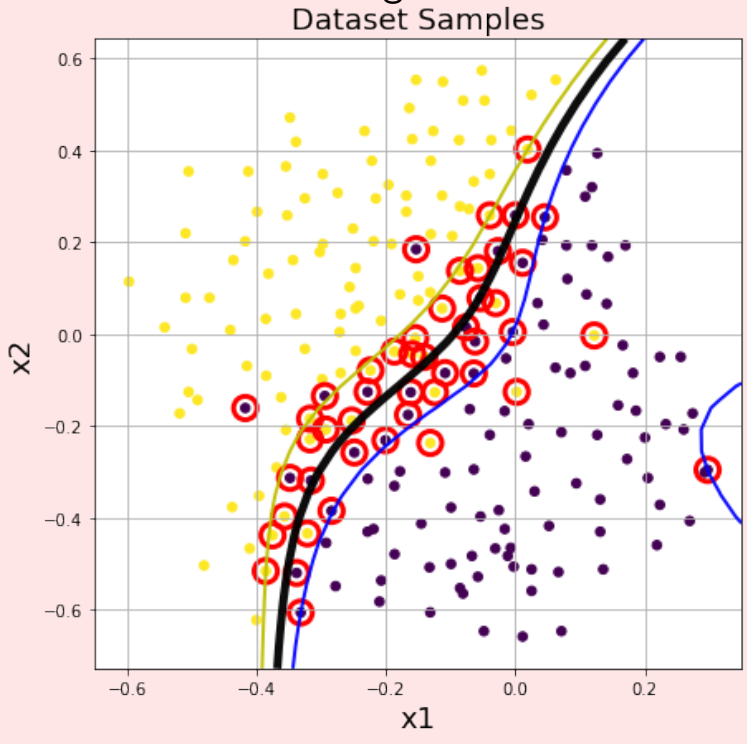
SVM with RBF kernel

Exercise 3: Hyperparameters values

Q1: Which set of parameters sounds good here? Explain why?

Solution: $C = 1000$ and $\gamma = 1$

The resulting decision boundary is smooth and it is a good compromise between large margin and some margin violations. It is more convenient for generalization.



Train classification accuracy: 0.934

Num. support vectors: 47

SVM with RBF kernel

Exercise 4: Cross-validation

The goal is to perform cross-validation to find the best hyperparameters (C and γ). The dataset was split into three subsets for cross-validation. Then, a search grid is implemented to find the hyperparameters that achieve the best classification accuracy (on test data).

Q1: Search for the best hyperparameters. Explain in a few words the process of 3-fold cross-validation that is applied on this example

Answer

Cross-validation is a technique to estimate the model (hyper)-parameters on limited data and avoid underfitting of overfitting. The data are split in k-folds, and, iteratively, one fold is used for testing and the rest k-1 folds are used for training (with different hyper-parameters). The hyper-parameters leading to the best (test) results on the aggregation of the different iterations are used as selected values.

SVM with RBF kernel

Exercise 4: Cross-validation

The goal is to perform cross-validation to find the best hyperparameters (C and γ). The dataset was split into three subsets for cross-validation. Then, a search grid is implemented to find the hyperparameters that achieve the best classification accuracy (on test data).

Q2: Report the best set of parameters (C_b and γ_b) that has been returned by the grid search. Given C_b, what is the range of γ_b values that are providing close results (e.g. the best results - 0.02)? What if we fix γ_b instead? What matters most in this case? Observing the distribution of data, what give you a hint on how to select the γ parameter?

Answer

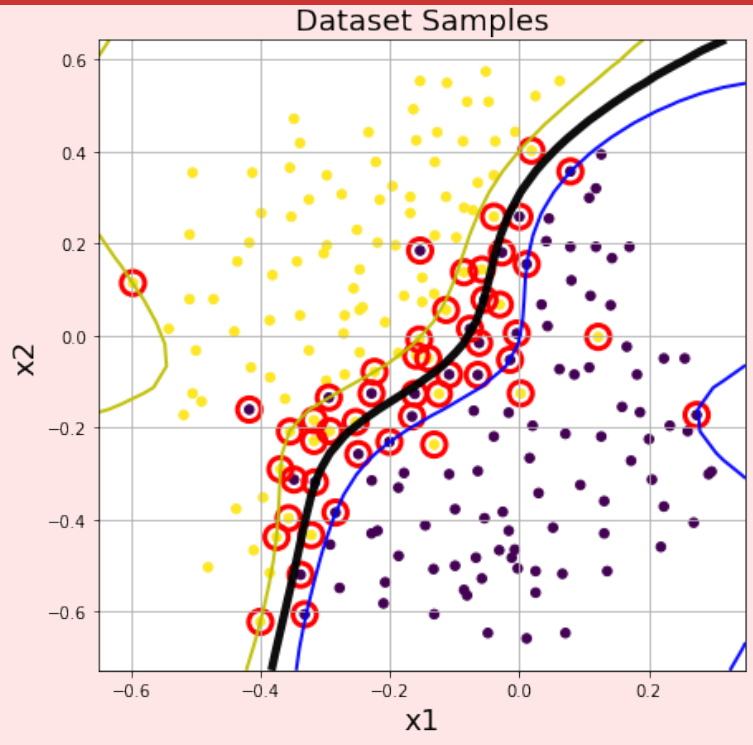
- Best obtained parameters: $C_b = 5$ and $\gamma_b = 5$, leading to the maximum cross-validation accuracy: 0.934.
- For $C_b = 5$ fixed, we get a range of γ from 5 to 25. For $\gamma_b = 5$ fixed, we get a range of C from 5 to 145. Therefore, the hyper-parameter γ is more important, which is to be expected as it provides the scale.
- In general, γ should be chosen by considering the density of the data points, or, in other words, the distance between the points in the dataset (e.g. average of the 5 closest distance between points). This is similar to the spectral clustering problem, which clusters points based on connectivity.

SVM with RBF kernel

Exercise 4: Cross-validation

Q3: Compute again the solution of exercise 2 with the obtained parameters. What can you conclude.

Solution:



The classification accuracy is 0.938; the number of support vectors is 50

Visually, the obtained result seems a good compromise between smoothness/margin size, and margin violations.