



Homework 1

Machine Learning and Deep Learning

Teaching Assistant:

Antonio D'Innocente

dinnocente@diag.uniroma1.it

Homework submission

- Individual work (**no groups**)
 - **3** Homeworks
 - Score \rightarrow **[0,2]** for each one
 - Total **score sufficient** for the admission to the final exam \rightarrow **[4,6]**
 - 4 \rightarrow no extra point
 - 5 \rightarrow +1 point
 - 6 \rightarrow +2 points
- } This holds **ONLY** if you submit the Project!
- You can keep your homework till the last exam session before the next class beginning (**1 year**)

Homework submission

- **Deadline** → **15 days** before the **first** exam of the session

Sessions of June-July: 15 days before the June exam

Homework submission

- **Uploading:** use the “Portale della didattica”

Submit a zip file named ID_MLDDL_homework1.zip (ID = matricola)

It should contain

- the report: a pdf file briefly describing the data, the logic of the methods and results with discussions
- the code

Note: **the report is *not* cut-and-paste the code in a pdf.**

You can copy some code lines in the report but only if they are presented together with the method explanation.

Homework 1: Nearest Neighbors, Linear SVM, SVM with RBF Kernel

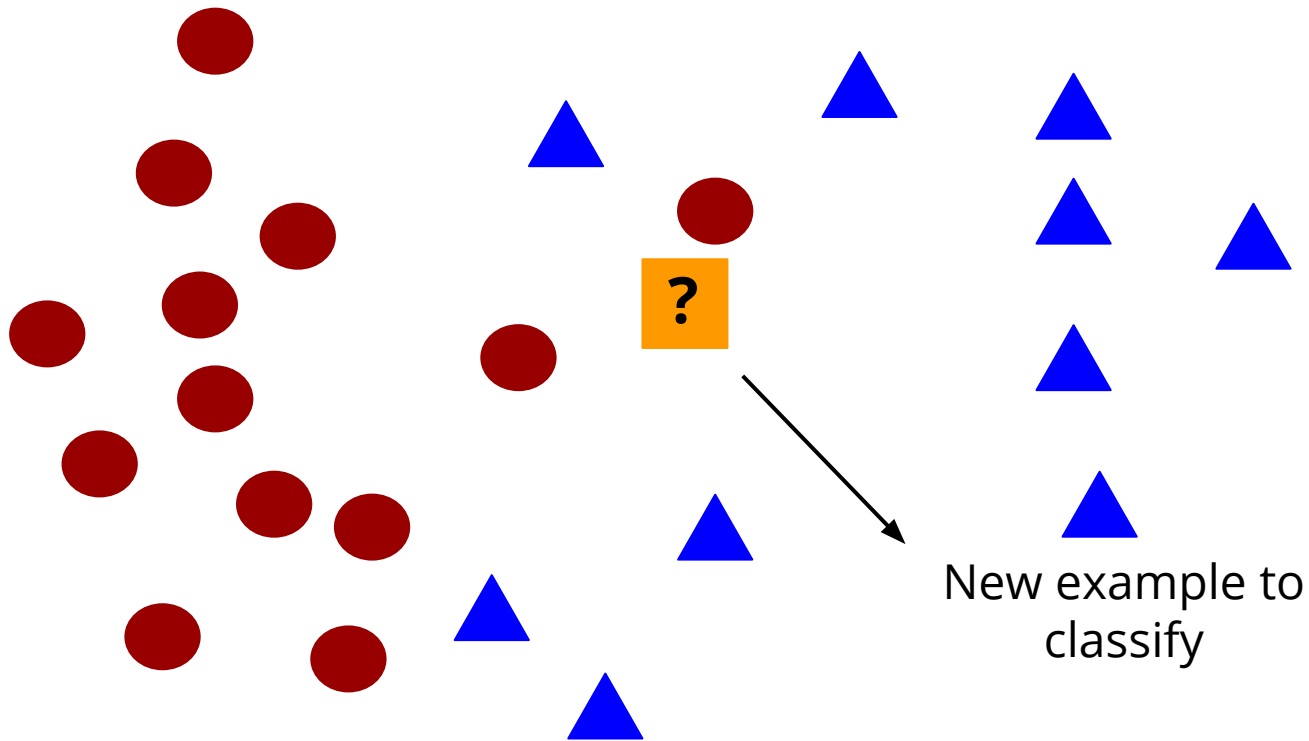
- Recap: NN, SVM, RBF Kernel
- Training procedure and Hyperparameter optimization
- Assignment

Recap: k-NN

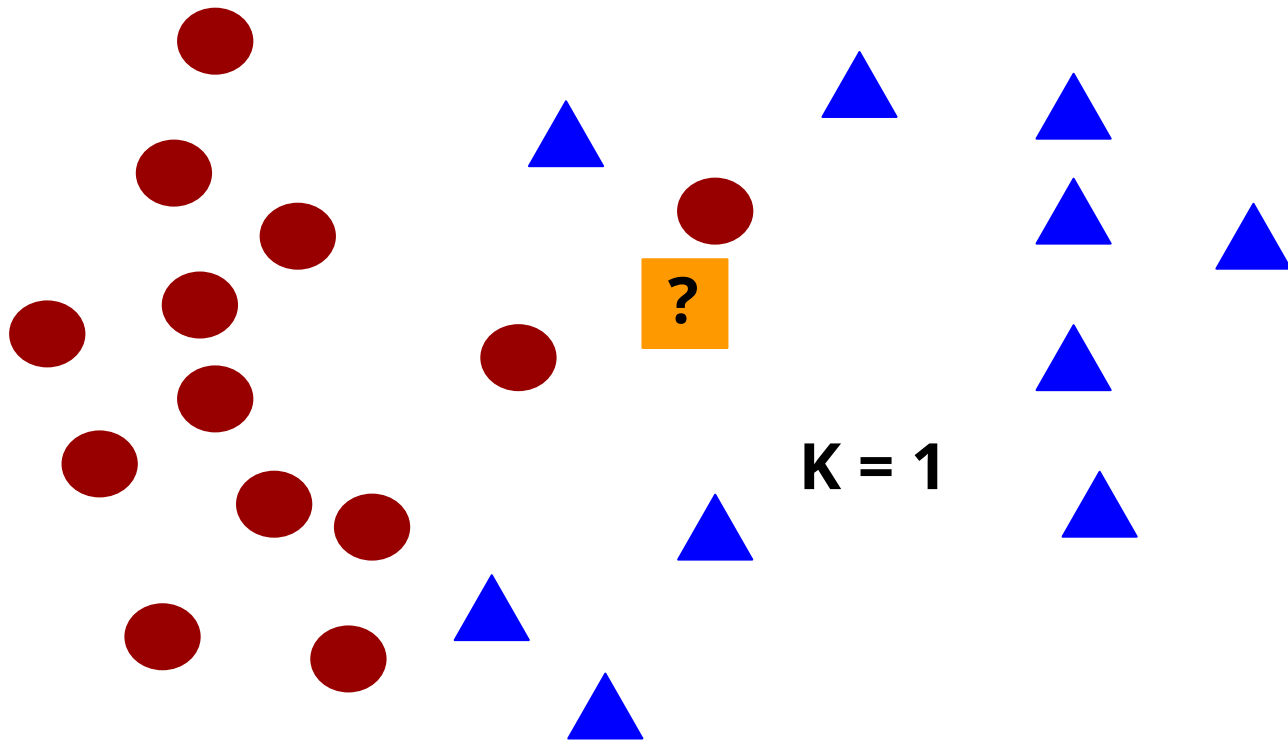
Recap: K-Nearest Neighbors

- Given a training set $S = (x_1, y_1), \dots, (x_m, y_m)$, k-NN generates a classifier $h_{k\text{-NN}}$ such that $h_{k\text{-NN}}(x)$ is the label y appearing in the majority of the k points $x_t \in S$ which are closest to x
- k-NN is a family of algorithms (one for each value of k)
- $k \leq m$

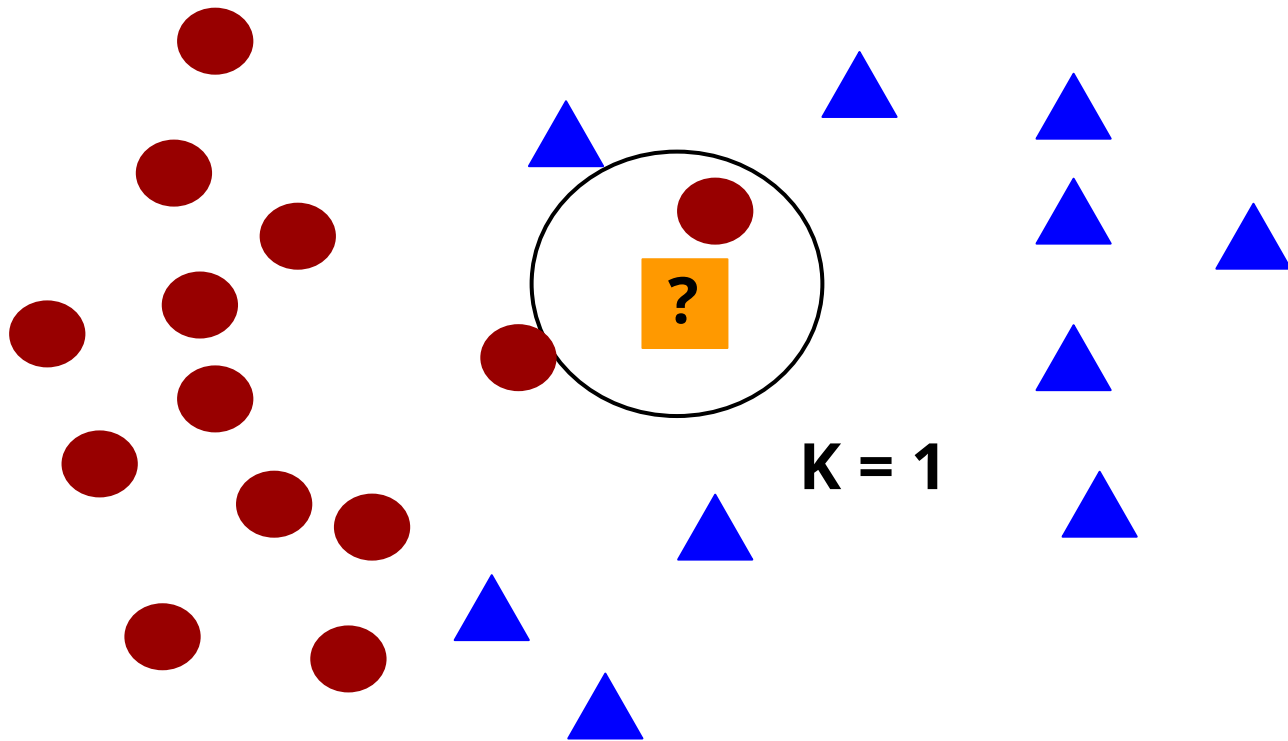
Recap: K-Nearest Neighbors



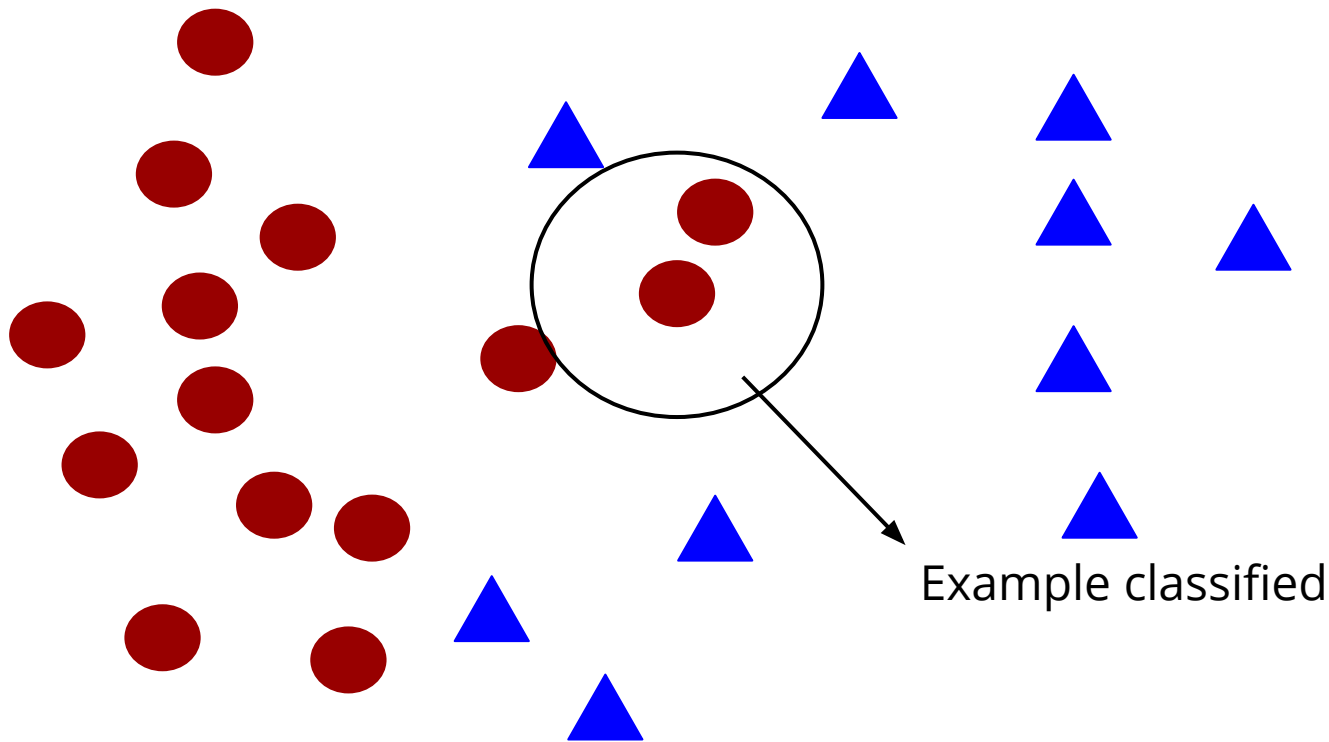
Recap: K-Nearest Neighbors



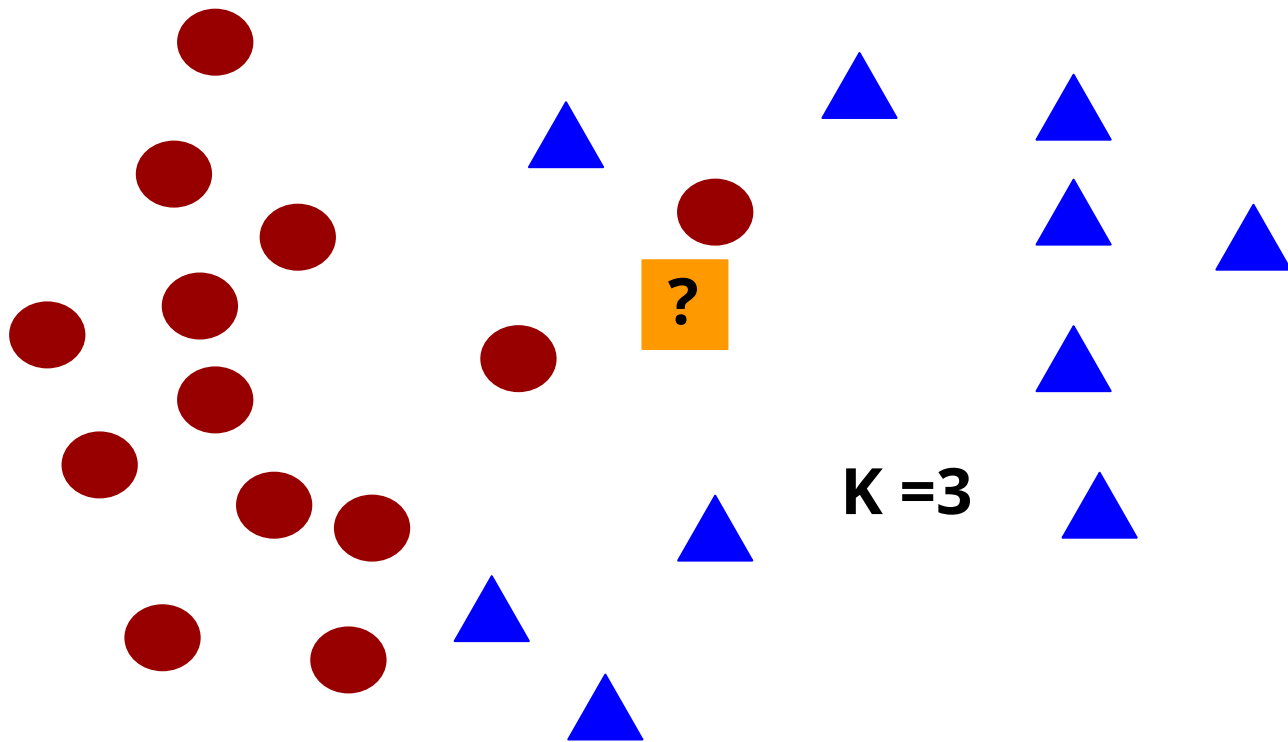
Recap: K-Nearest Neighbors



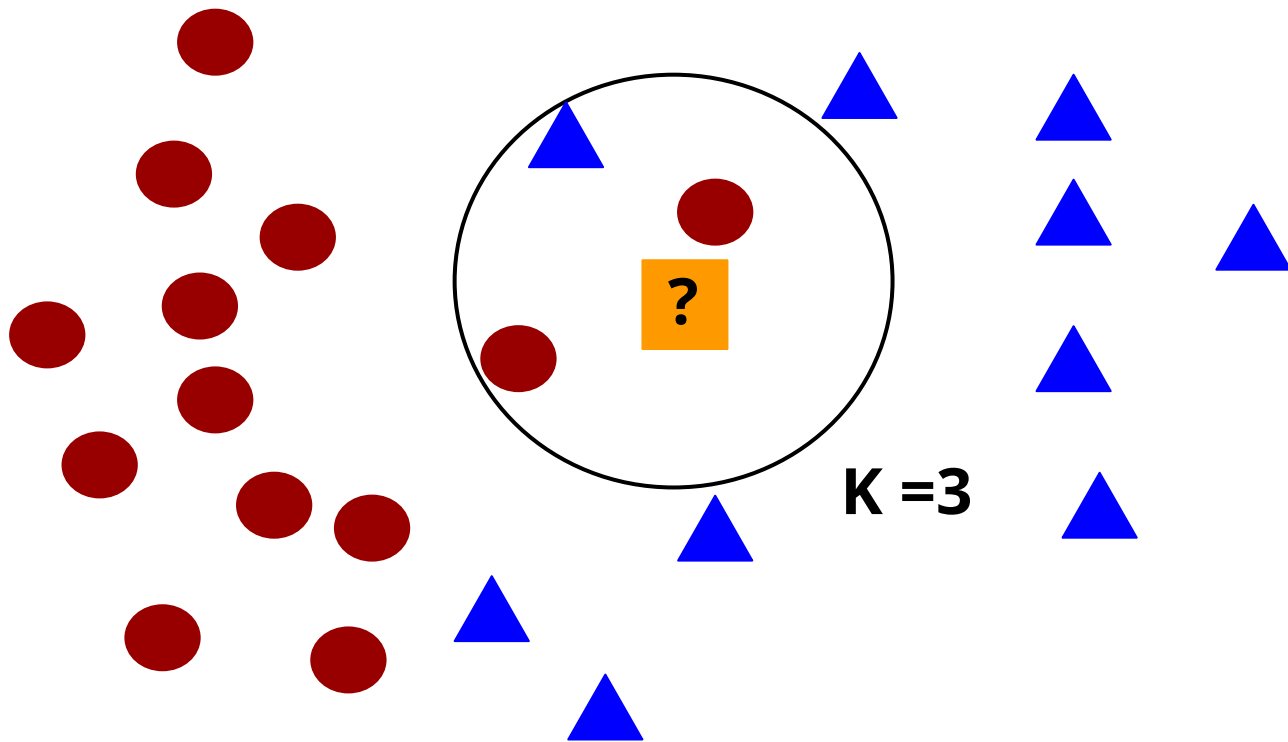
Recap: K-Nearest Neighbors



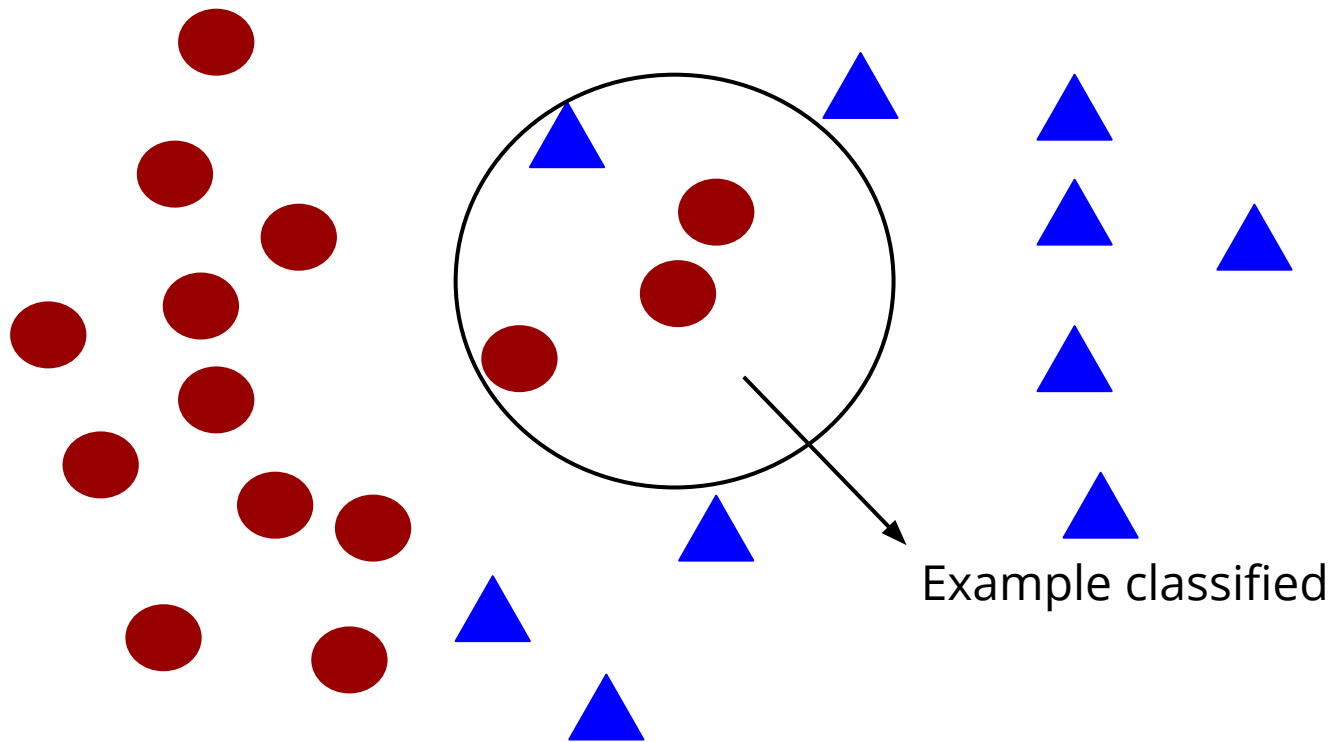
Recap: K-Nearest Neighbors



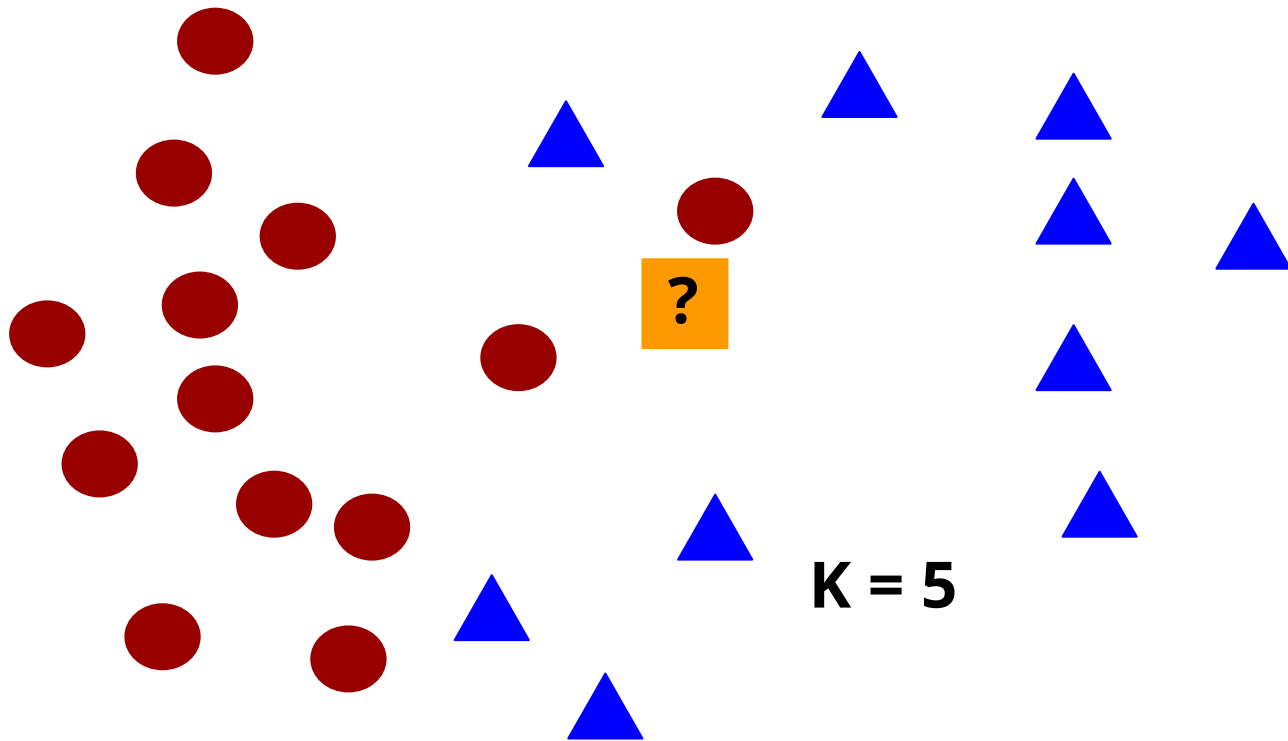
Recap: K-Nearest Neighbors



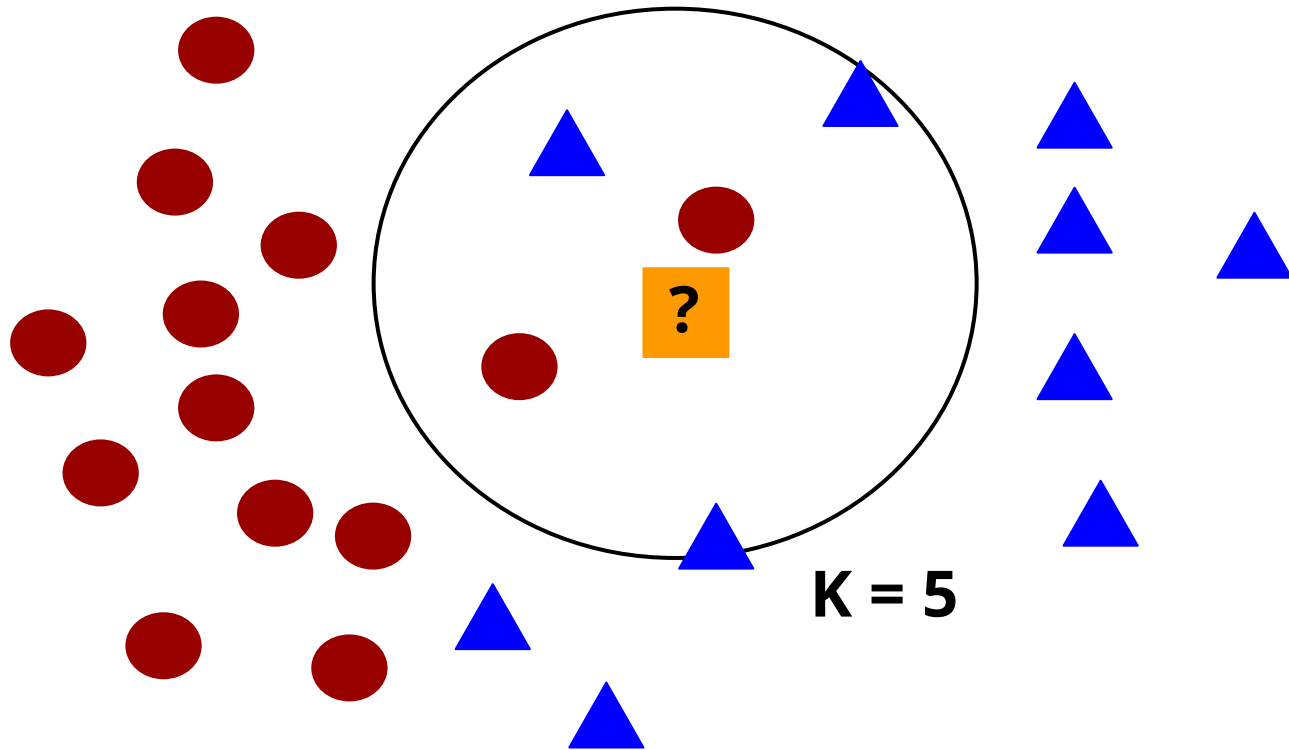
Recap: K-Nearest Neighbors



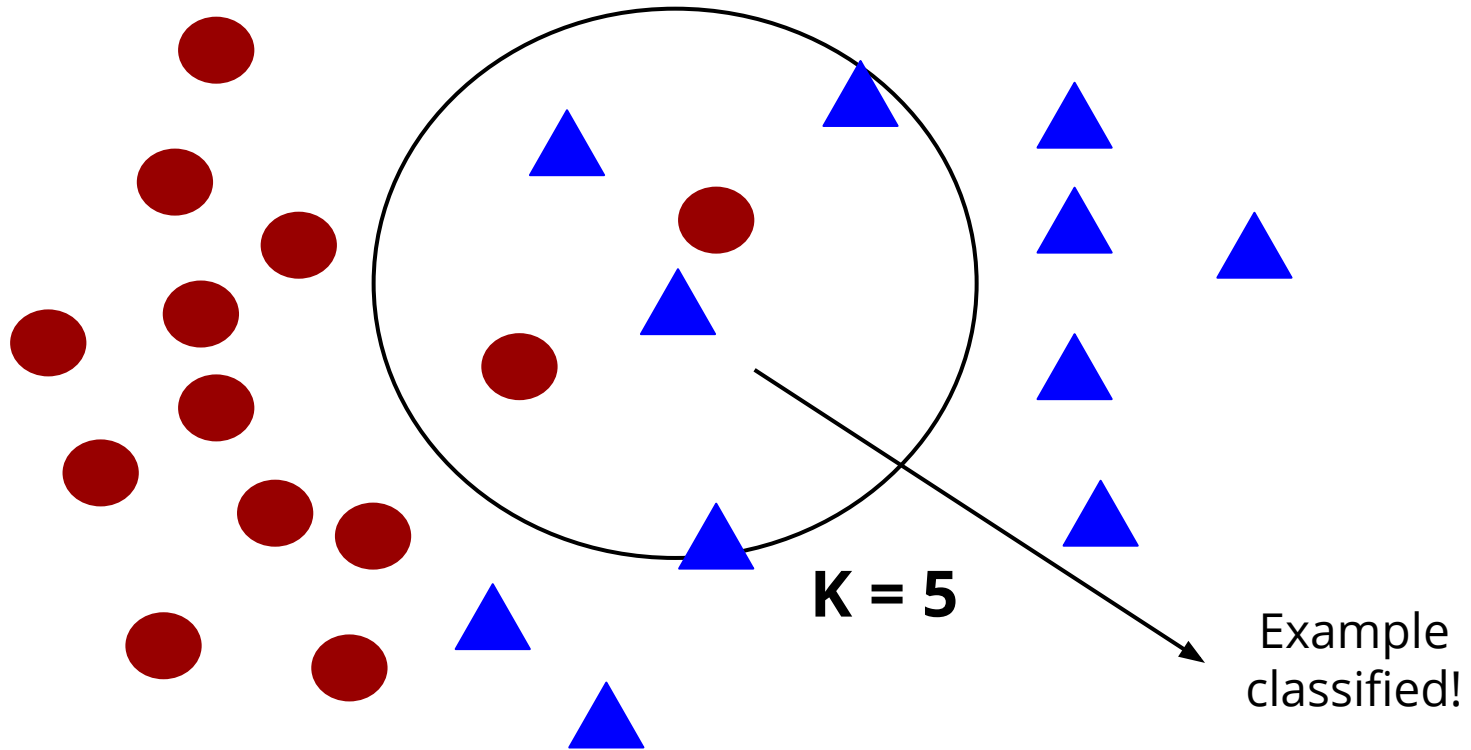
Recap: K-Nearest Neighbors



Recap: K-Nearest Neighbors



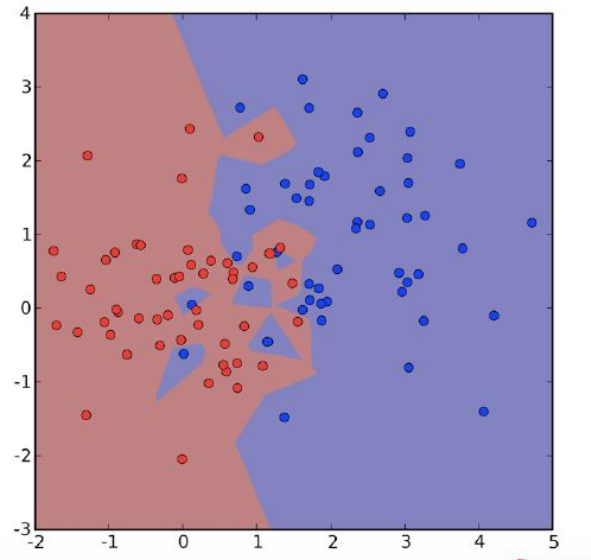
Recap: K-Nearest Neighbors



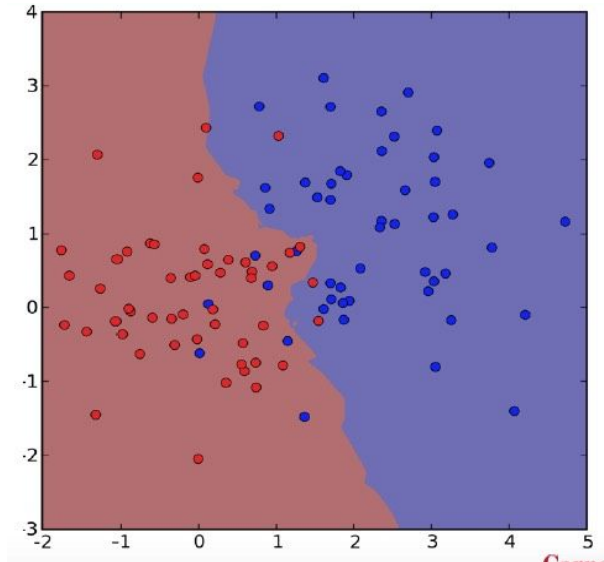
K-NN - Hyper Parameter

K is the number of neighbors used for the voting.

K = 1



K = 4



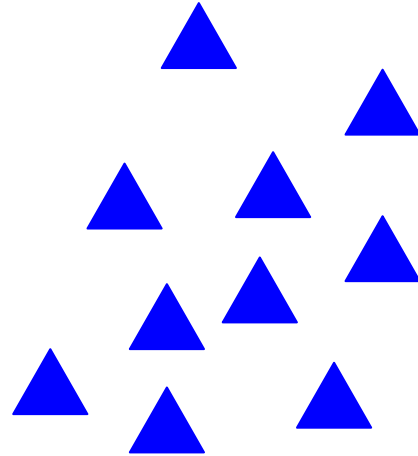
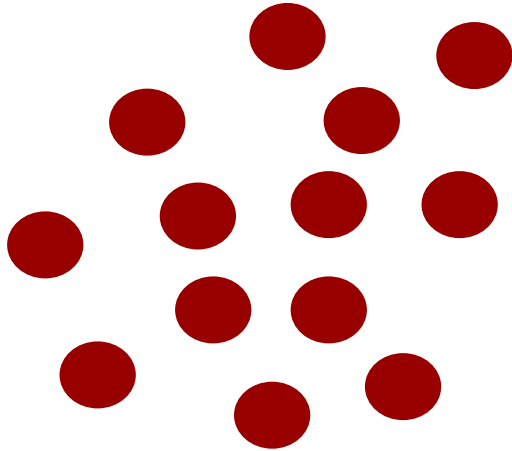
Recap: Linear SVM

Recap: Linear SVM

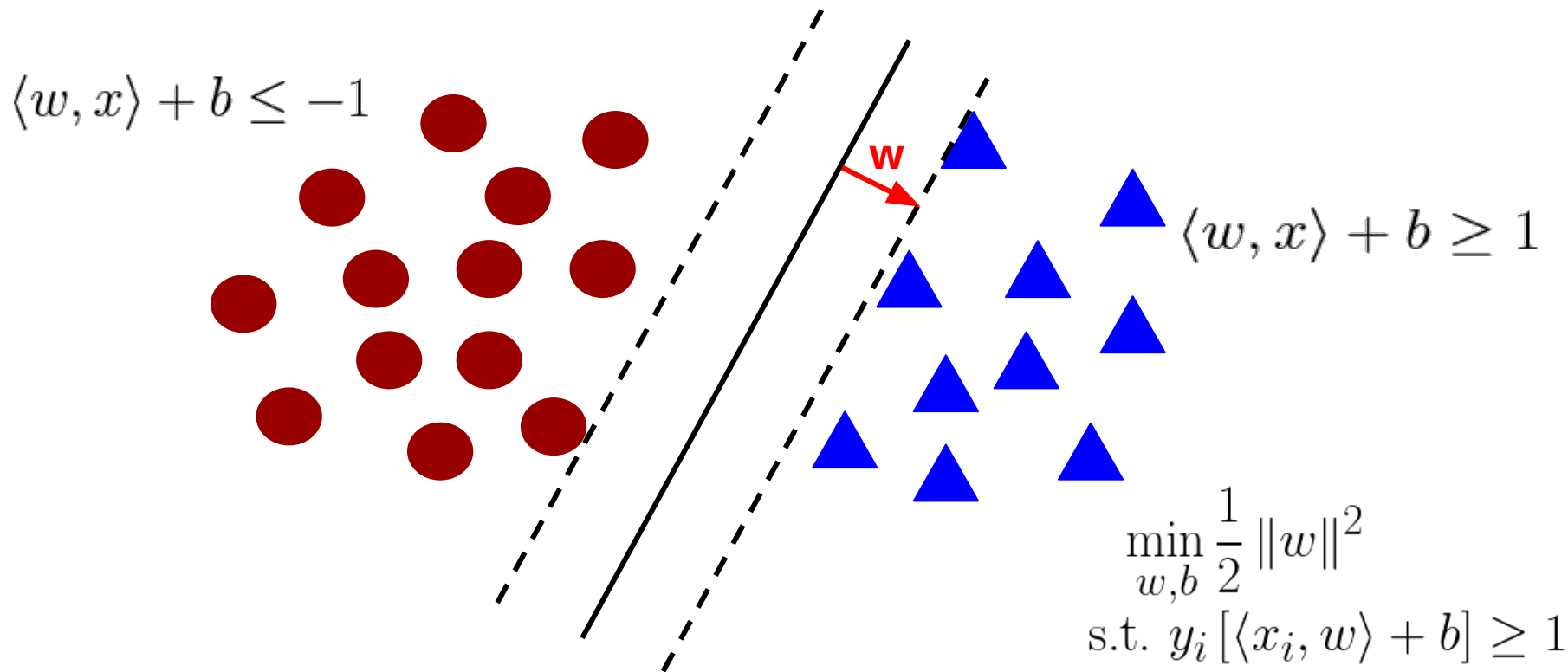
- Given a linearly separable training set $S = (x_1, y_1), \dots, (x_m, y_m)$ with $y \in \{+1, -1\}$ SVM outputs the linear classifier corresponding to the unique solution of:

$$\begin{aligned} \min_{w, b} \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y_i [\langle x_i, w \rangle + b] \geq 1 \end{aligned}$$

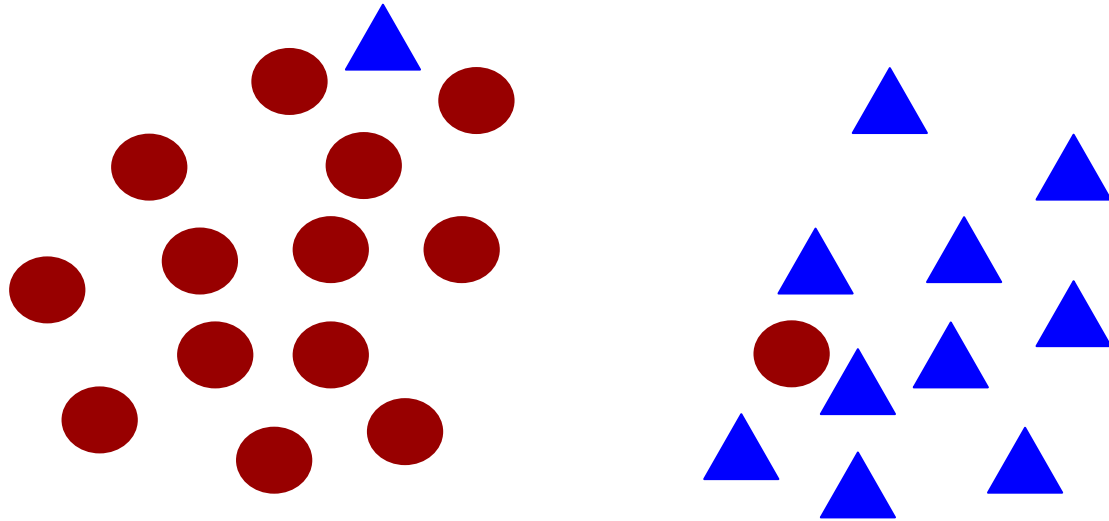
Recap: Linear SVM



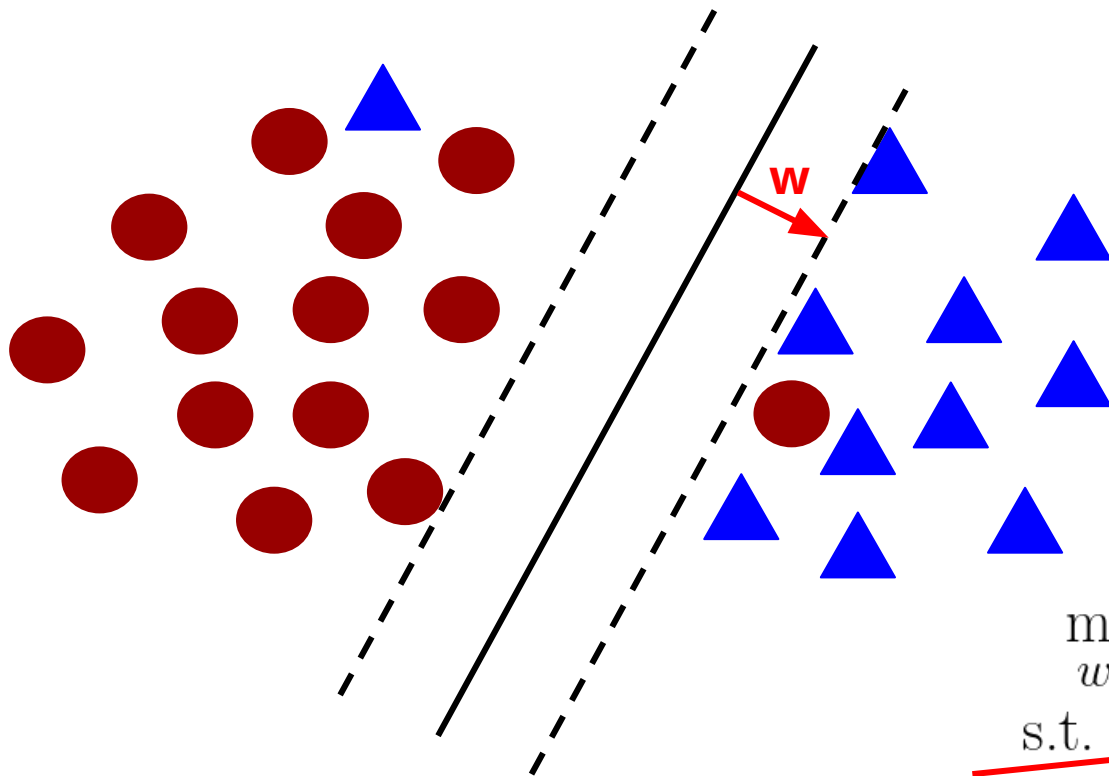
Recap: Linear SVM - Hard margin problem



Recap: Linear SVM - Soft margin problem

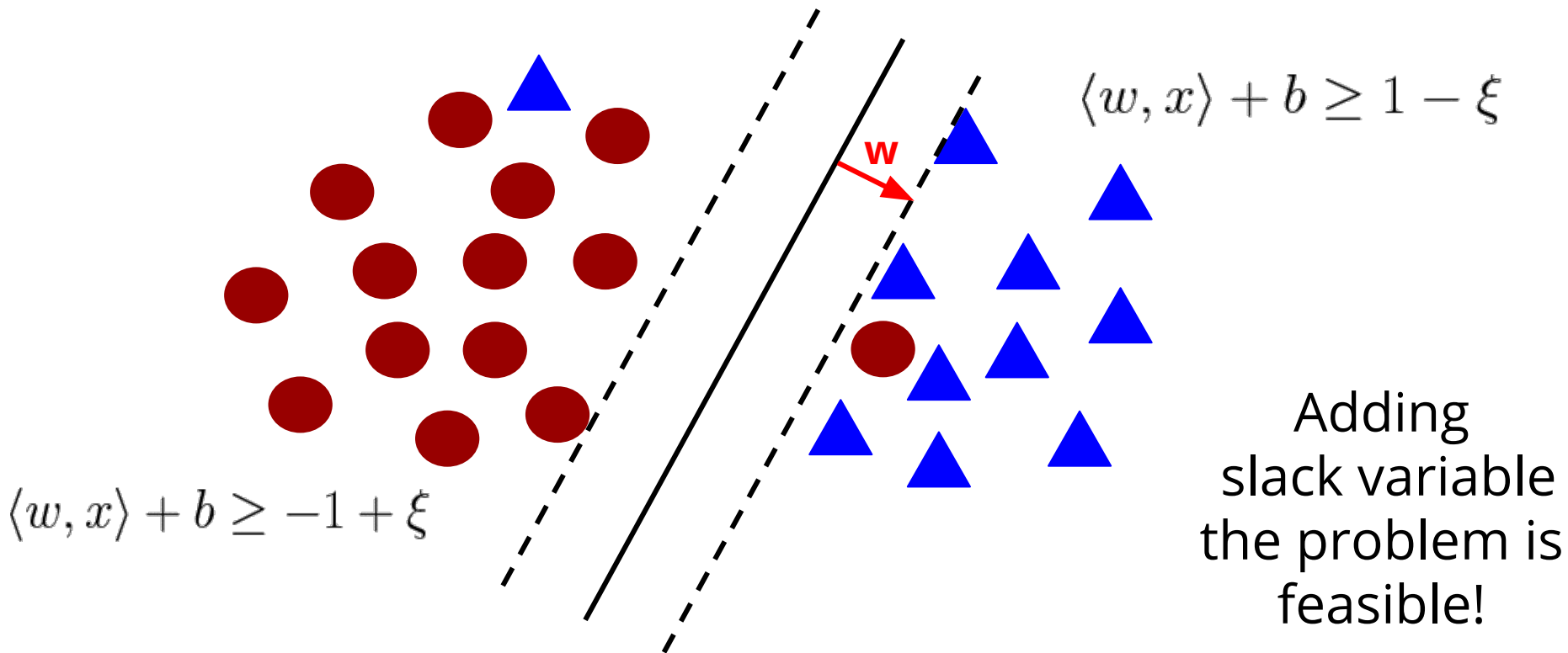


Recap: Linear SVM - Soft margin problem



$$\begin{aligned} \min_{w, b} \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y_i [\langle x_i, w \rangle + b] \geq 1 \end{aligned}$$

Recap: Linear SVM - Soft margin problem



Recap: Linear SVM

Hard margin problem

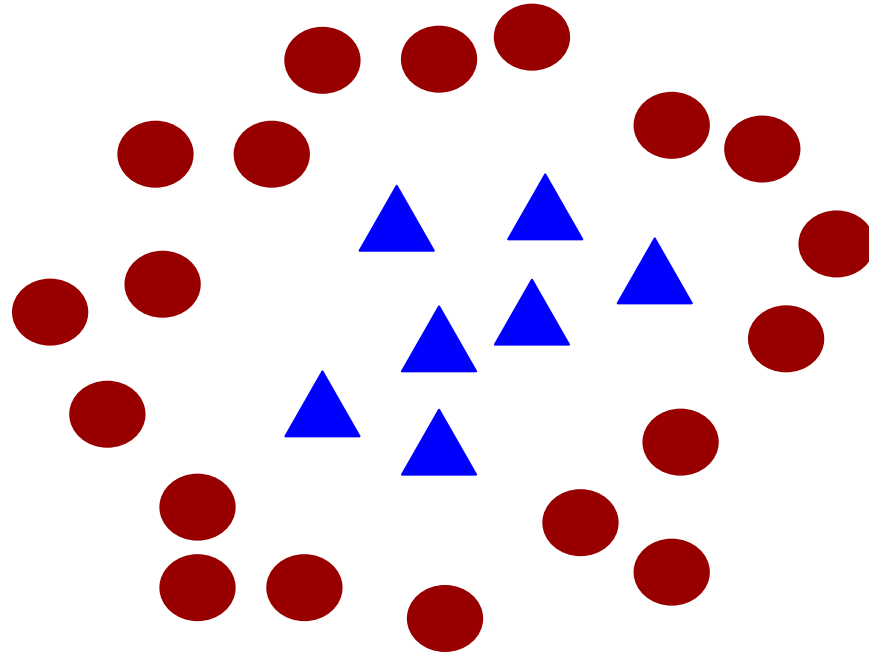
$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y_i [\langle x_i, w \rangle + b] \geq 1 \end{aligned}$$

Soft margin problem

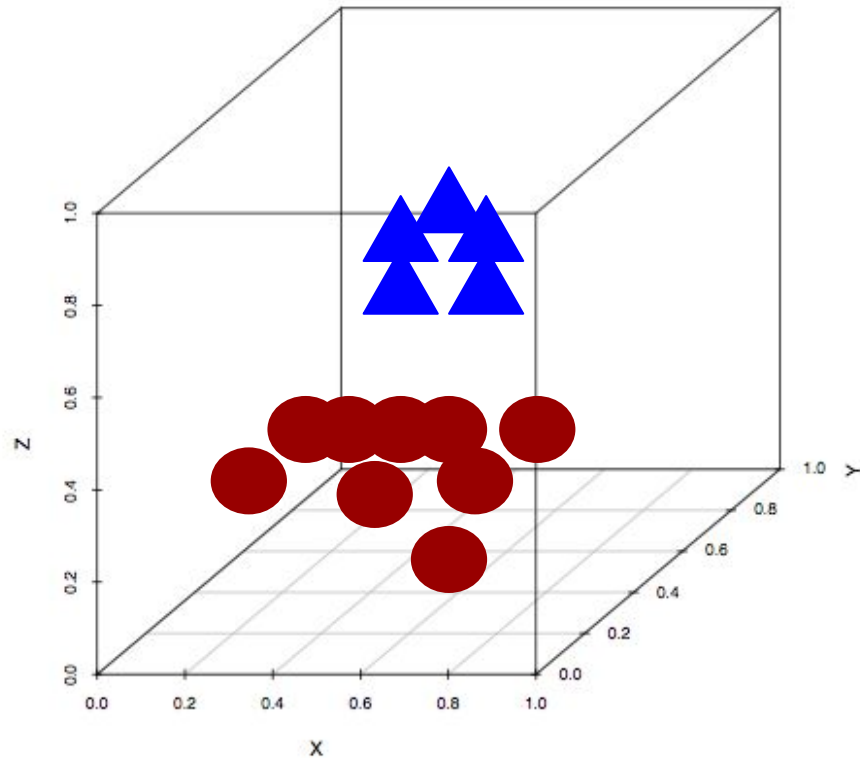
$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \\ \text{s.t.} \quad & y_i [\langle x_i, w \rangle + b] \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \end{aligned}$$

Recap: RBF Kernel

Recap: Kernel trick - SVM



Recap: Kernel trick - SVM



We can separate each data point by projecting it into an **higher dimension**

Recap: Kernel trick - SVM

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i$$

s.t. $y_i [\langle x_i, w \rangle + b] \geq 1 - \xi_i$ and $\xi_i \geq 0$

Recap: Kernel trick - SVM

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i$$

s.t. $y_i [\langle x_i, w \rangle + b] \geq 1 - \xi_i$ and $\xi_i \geq 0$



$\Phi(x_i)$

Hard to compute!

Recap: Kernel trick - SVM

..so we use the **dual** soft margin problem

$$\begin{aligned} \max_{\alpha} & -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_i \alpha_i \\ \text{s.t.} & \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \in [0, C] \end{aligned}$$

Recap: Kernel trick - SVM

..so we use the **dual** soft margin problem

$$\begin{aligned} \max_{\alpha} & -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_i \alpha_i \\ \text{s.t.} & \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \in [0, C] \end{aligned}$$

to be able to use the
KERNEL function

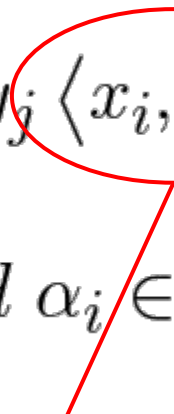
$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

Recap: Kernel trick - SVM

..so we use the **dual** soft margin problem

$$\max_{\alpha} -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_i \alpha_i$$

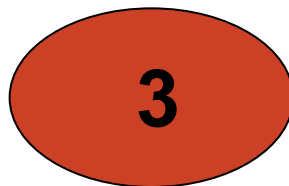
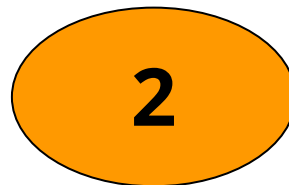
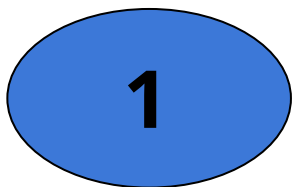
s.t. $\sum_i \alpha_i y_i = 0$ and $\alpha_i \in [0, C]$



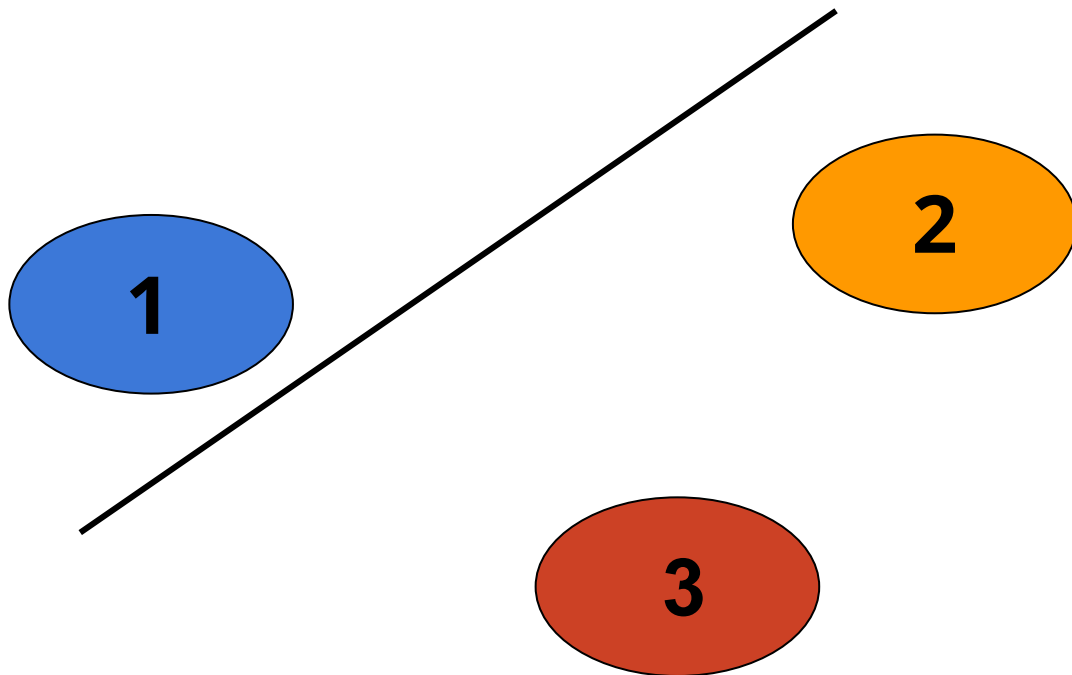
we will use the RBF
Kernel

$$k(x_i, x_j) = e^{-\gamma(x_i - x_j)^2}$$

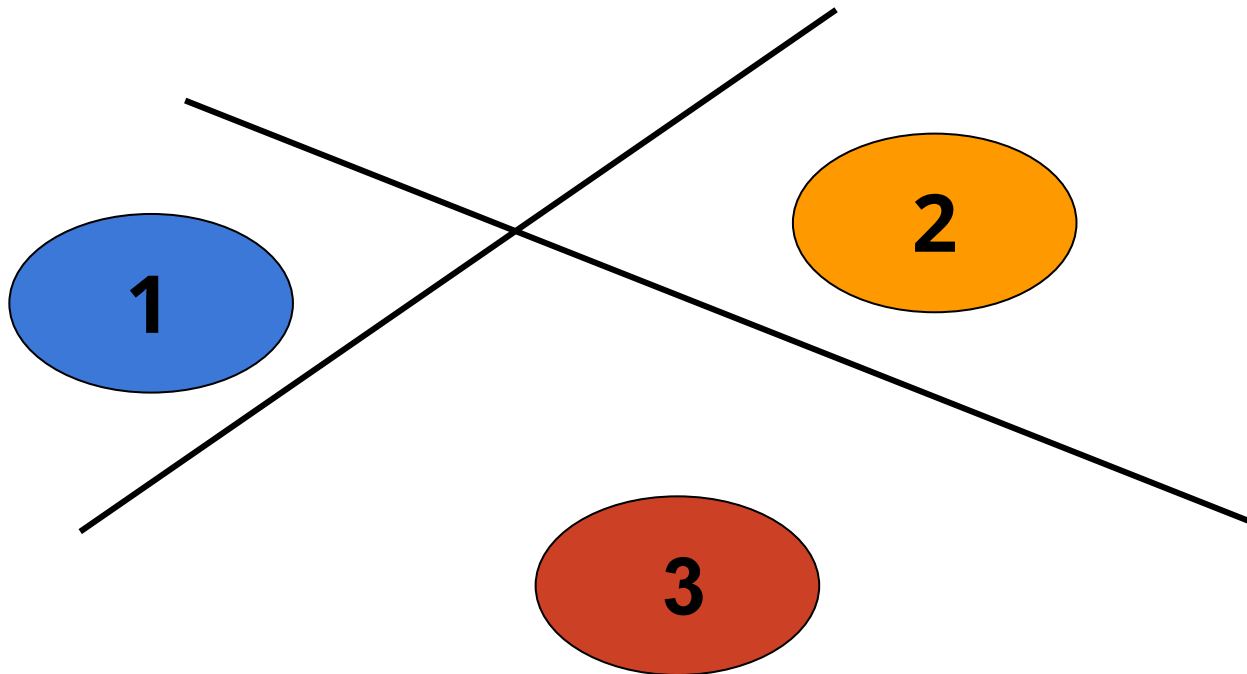
SVM - Multiclass: one vs all



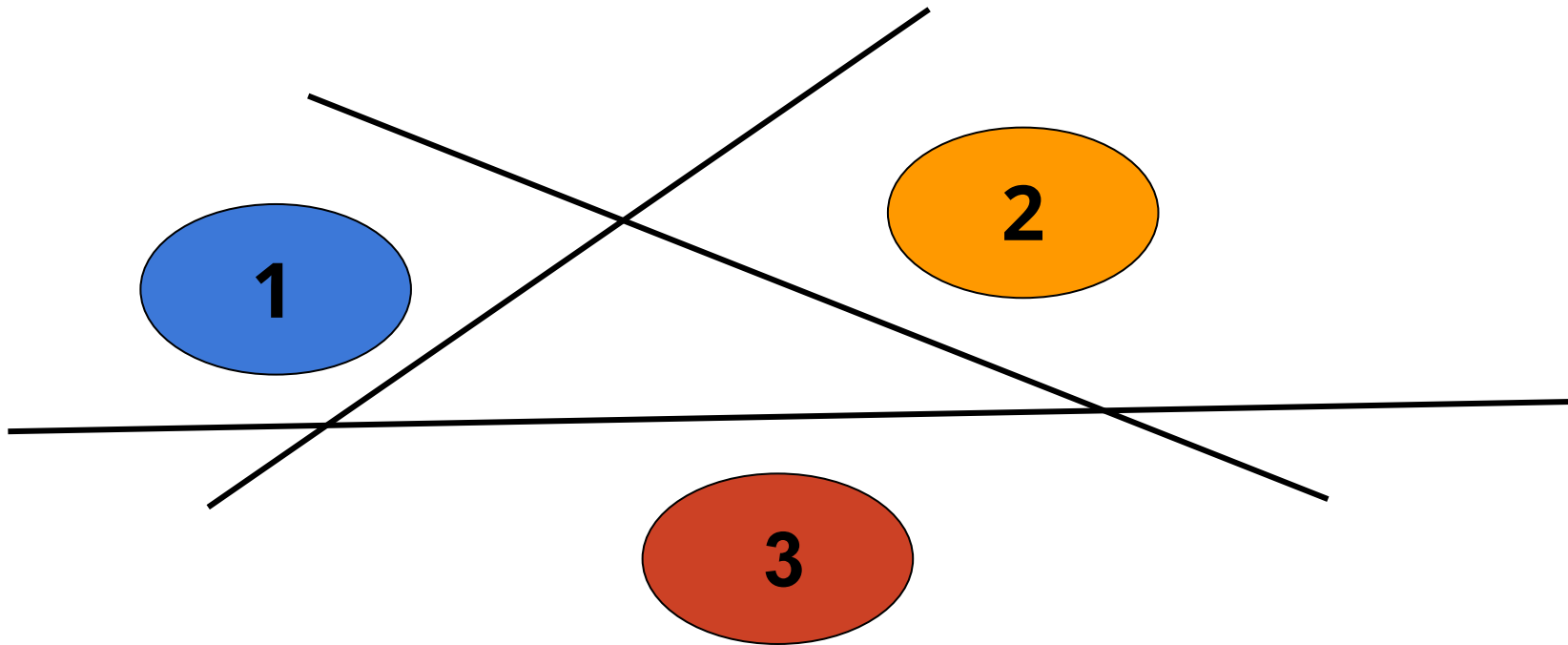
SVM - Multiclass: one vs all



SVM - Multiclass: one vs all



SVM - Multiclass: one vs all



Hyperparameters

SVM - Hyper Parameters

Soft margin problem:

$$\min_{w,b} \frac{1}{2} \|w\|^2 + \underline{C} \sum_i \xi_i$$

s.t. $y_i [\langle x_i, w \rangle + b] \geq 1 - \xi_i$ and $\xi_i \geq 0$

- **C**

SVM - Hyper Parameters

Soft margin problem:

$$\min_{w,b} \frac{1}{2} \|w\|^2 + \underline{C} \sum_i \xi_i$$

s.t. $y_i [\langle x_i, w \rangle + b] \geq 1 - \xi_i$ and $\xi_i \geq 0$

- **C**

***C** is the cost of misclassification*

SVM - Hyper Parameters

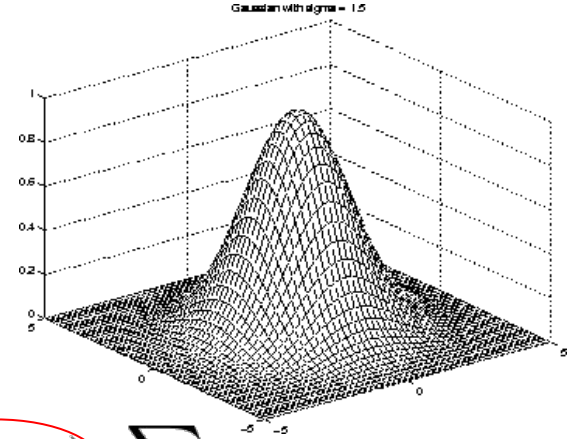
Kernel RBF:

- **C**

$$\begin{aligned} \max_{\alpha} & -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) + \sum_i \alpha_i \\ \text{s.t.} & \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \in [0, \underline{C}] \end{aligned}$$

SVM - Hyper Parameters

γ is inversely proportional to the variance of the Gaussian



Kernel RBF:

- C
- γ

$$\max_{\alpha} -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) + \sum_i \alpha_i$$

$$s.t. \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \in [0, \underline{C}]$$

$$k(x_i, x_j) = e^{-\underline{\gamma}(x_i - x_j)^2}$$

SVM - Hyper Parameters

Soft margin problem:

- **C**

$$\min_{w,b} \frac{1}{2} \|w\|^2 + \underline{C} \sum_i \xi_i$$

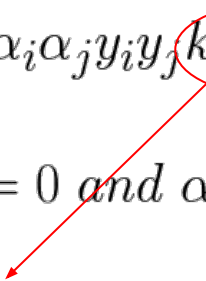
s.t. $y_i [\langle x_i, w \rangle + b] \geq 1 - \xi_i$ and $\xi_i \geq 0$

Kernel RBF:

- **C**
- **γ**

$$\max_{\alpha} -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) + \sum_i \alpha_i$$

s.t. $\sum_i \alpha_i y_i = 0$ and $\alpha_i \in [0, \underline{C}]$

$$k(x_i, x_j) = e^{-\underline{\gamma}(x_i - x_j)^2}$$


How to choose the Hyper Parameters?

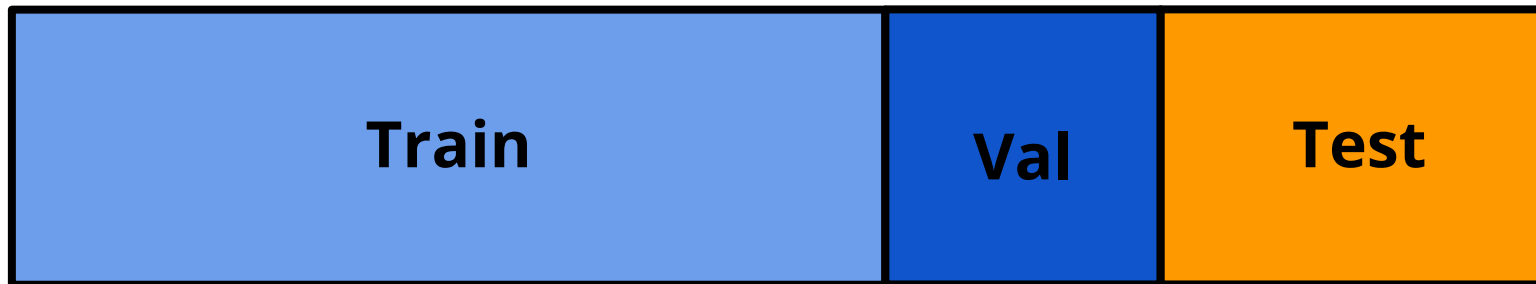
Training procedure



Dataset

To analyze the performance and evaluate a model you need to divide the dataset into **3 splits**

Training procedure

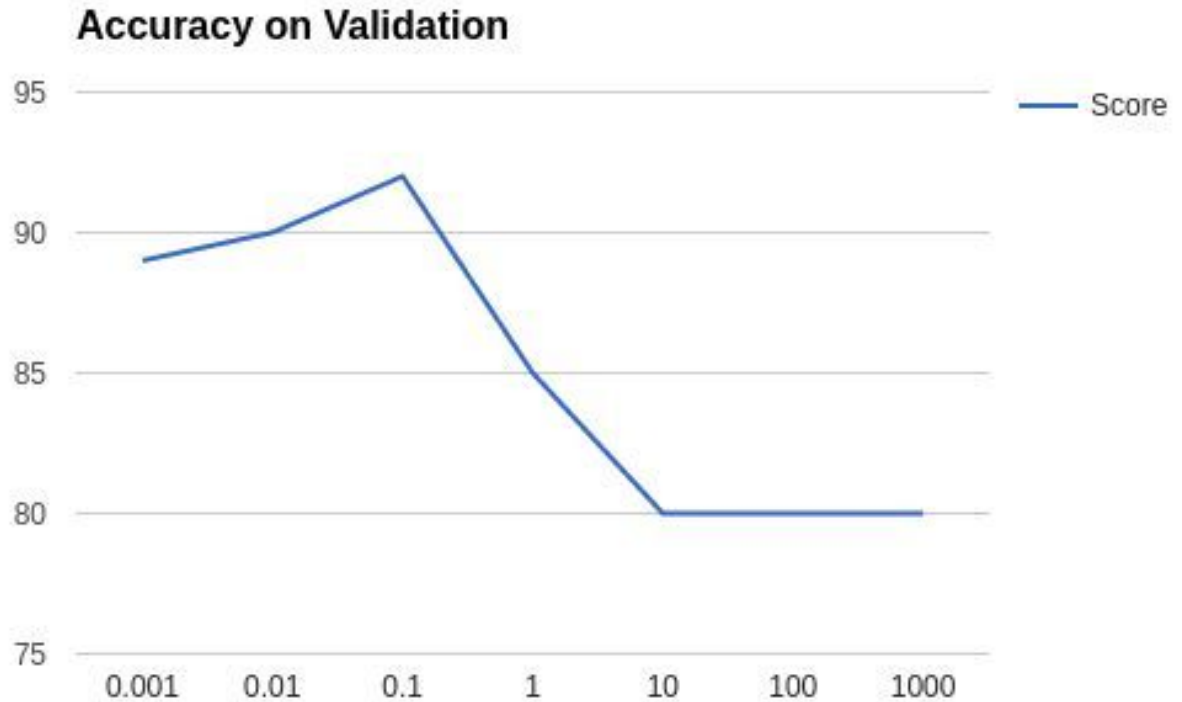


1. Train the model using only the **training set** with the **parameters chosen**
2. Evaluate the performance of the model on the **validation set**
3. Choose others parameters and repeat 1. and 2. until you find the parameters that work best on the validation set.

At the end you can evaluate the model on the **testing set** but you cannot change the parameters in function of the performance on the testing set!

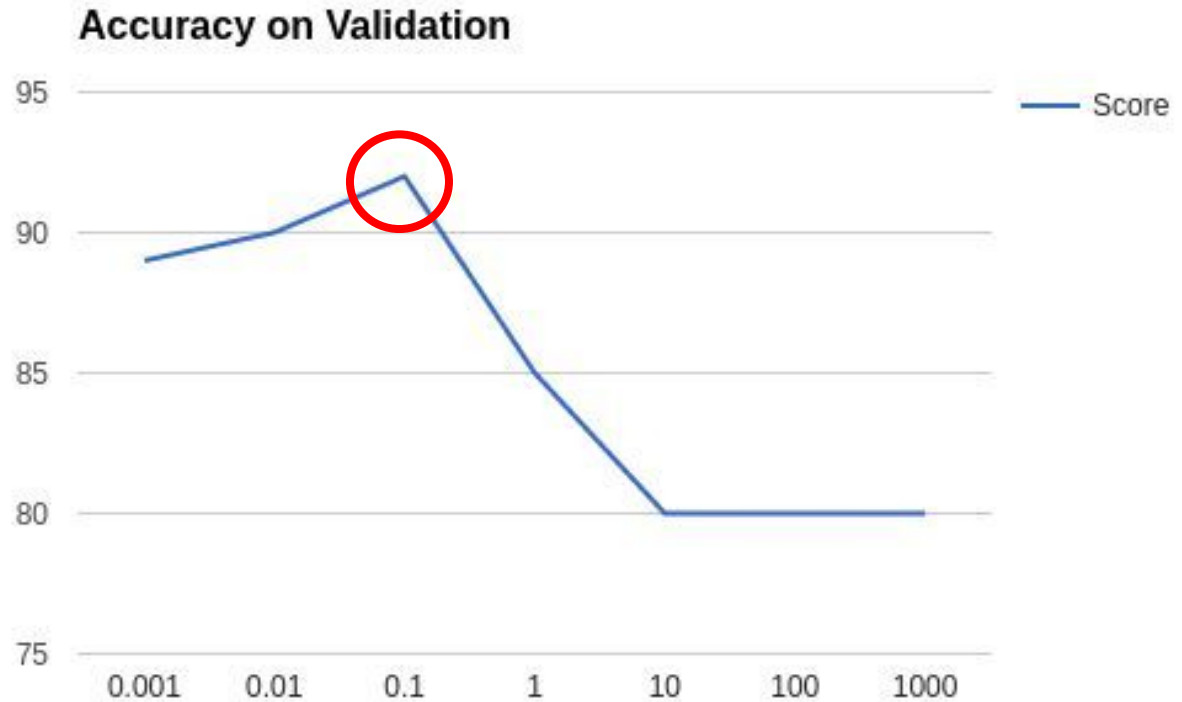
Training procedure - Example

Optimization of **C**



Training procedure - Example

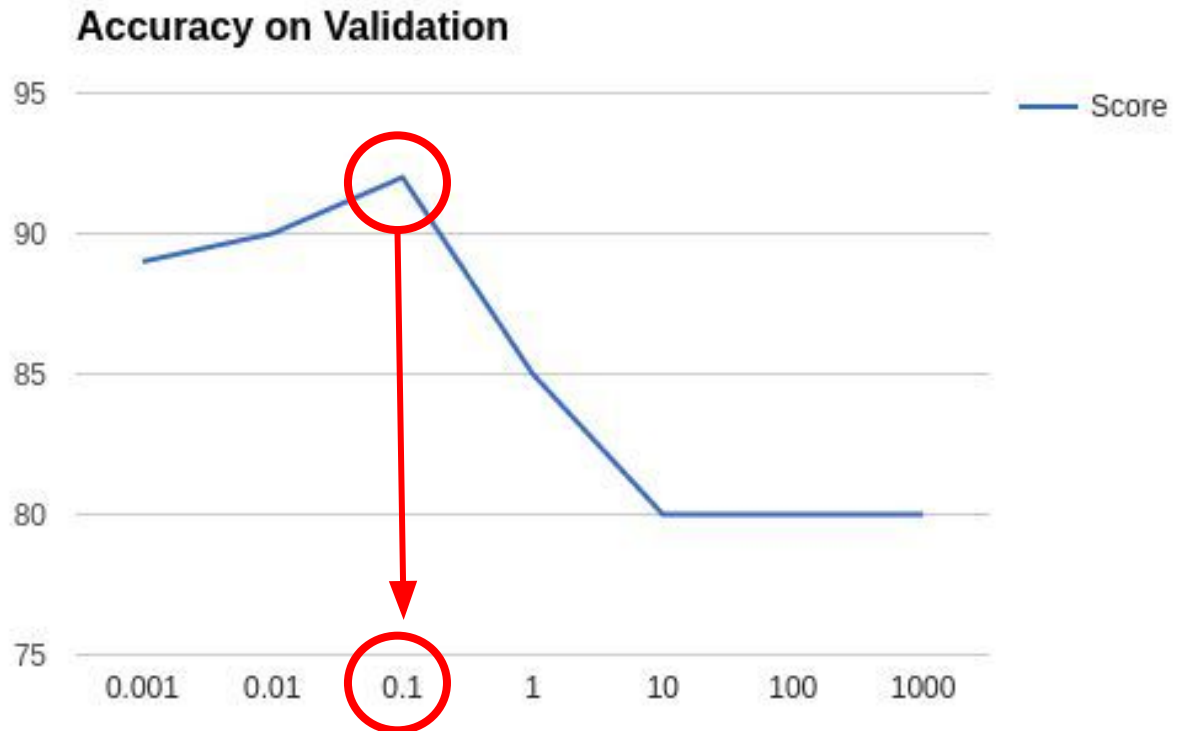
Optimization of **C**



Training procedure - Example

Optimization of **C**

Best C found!



Training procedure - Example - Grid Search

Optimization of **C** and **γ**

| Gamma \ C | 10^{-2} | 10^{-1} | 10^0 | 10^1 |
|-----------|-----------|-----------|--------|--------|
| 10^{-9} | 66% | 75% | 60% | 60% |
| 10^{-7} | 48% | 67% | 78% | 79% |
| 10^{-5} | 80% | 82% | 90% | 85% |
| 10^{-3} | 45% | 66% | 78% | 74% |

Training procedure - Example - Grid Search

Optimization of **C** and **γ**

**Best C and γ
found!**

| Gamma \ C | 10^{-2} | 10^{-1} | 10^0 | 10^1 |
|-----------|-----------|-----------|--------|--------|
| 10^{-9} | 66% | 75% | 60% | 60% |
| 10^{-7} | 48% | 67% | 78% | 79% |
| 10^{-5} | 80% | 82% | 90% | 85% |
| 10^{-3} | 45% | 66% | 78% | 74% |

Training procedure - K-Fold Cross Validation

..for a more precise evaluation of the model.

K = 4

Round 1



Round 2



Round 3



Round 4



Final Accuracy = Average(Round 1, Round 2, Round 3, Round 4)

Assignment

Homework - K-Nearest Neighbors

1. Load **Wine** dataset (scikit library)
2. Select the first two attributes for a 2D representation of the image.
3. Randomly split data into train, validation and test sets in proportion 5:2:3
4. For **K = [1,3, 5,7]**:
 - a. Apply **K-Nearest Neighbors**
 - b. Plot the data and the decision boundaries
 - c. Evaluate the method on the validation set
5. Plot a graph showing how the accuracy on the validation set varies when changing **K**
6. How the boundaries change? Why?
7. Use the best value of **K** and evaluate the model on the **test set**.
How well does it works?

Homework - Linear SVM

8. For $\mathbf{C} = [0.001, 0.01, 0.1, 1, 10, 100, 1000]$:
 - a. Train a **linear SVM** on the training set
 - b. Plot the data and the decision boundaries
 - c. Evaluate the method on the validation set
9. Plot a graph showing how the accuracy on the validation set varies when changing \mathbf{C}
10. How the boundaries change? Why?
11. Use the best value of \mathbf{C} and evaluate the model on the **test set**.
How well does it work?

Homework - RBF Kernel

12. Repeat point 8. (train, plot, etc..), but this time use an **RBF kernel**
13. Evaluate the best **C** on the **test set**
14. Are there any differences compared to the linear kernel? How are the boundaries different?
15. Perform a grid search of the best parameters for an RBF kernel: we will now tune both **gamma** and **C** at the same time. Select an appropriate range for both parameters. Train the model and score it on the validation set. Evaluate the best parameters on the test set. Plot the decision boundaries.

Homework - K-Fold

16. Merge the training and validation split. You should now have 70% training and 30% test data.
17. Repeat the grid search for **gamma** and **C** but this time perform 5-fold validation.
18. Evaluate the parameters on the test set. Is the final score different? Why?

Extra : only once you finish the main requests

19. Discuss the difference between KNN and SVM
20. Try also with different pairs of attributes

In practice..

You should use the **scikit-learn** library:

<https://scikit-learn.org/stable/>

In particular, check these docs for some help:

- <https://scikit-learn.org/stable/modules/neighbors.html#neighbors>
- <https://scikit-learn.org/stable/modules/svm.html>
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

Now it's your turn, try!

