Introduction To Machine Learning

UCL
Université catholique de Louvain

PRESENTATION LAYOUT

- Introduction and Machine Learning Principles
- Examples of Classifiers
- How to Perform Experiments
- Metrics
- Suggested Readings

Machine learning studies computer algorithms for learning to do stuff.
(Schapire)

A computer program is set to learn from an experience E with respect to some task T and some performance measure P if its performance on T as measured by P improves with experience E.
(Mitchell)

Problem Description

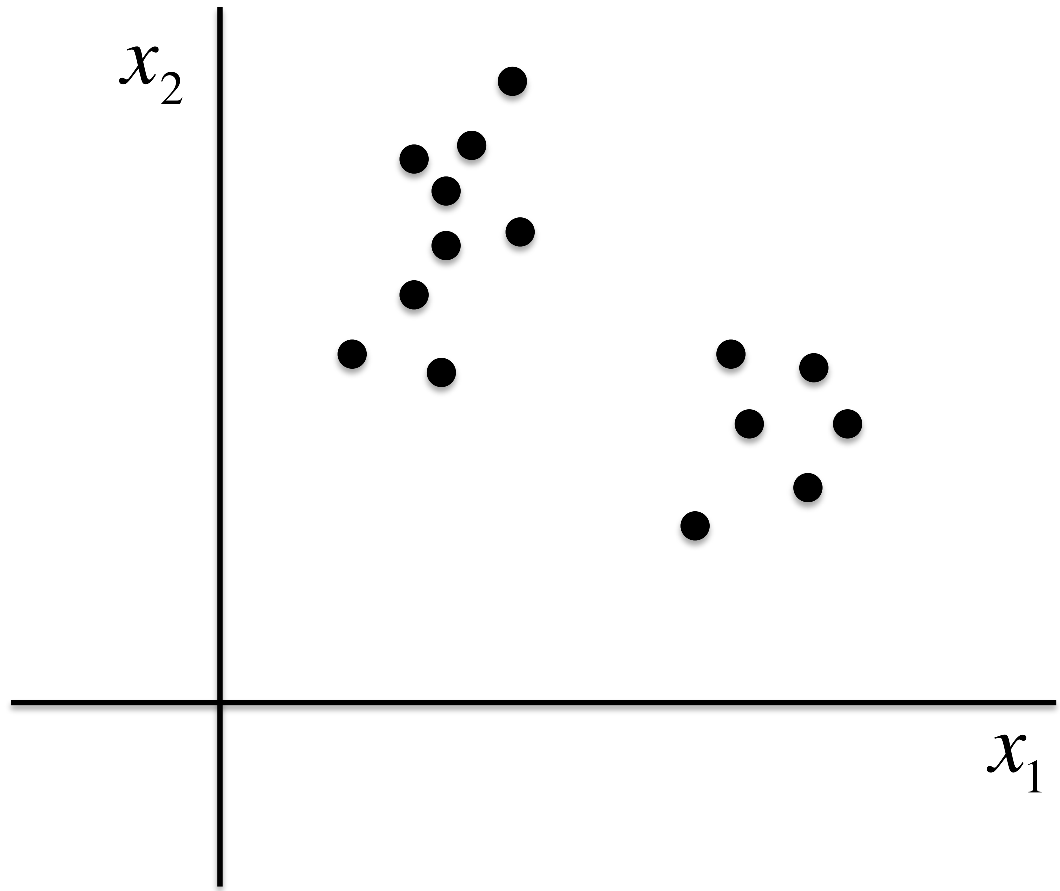
$$\bar{x} = (x_1, \ldots, x_m)^T$$

Sample

$$\{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_N\}$$

Dataset

$$\bar{x} = (x_1, x_2)^T$$



PROBLEM DESCRIPTION

$$\bar{x} = (x_1, \ldots, x_m)^T$$

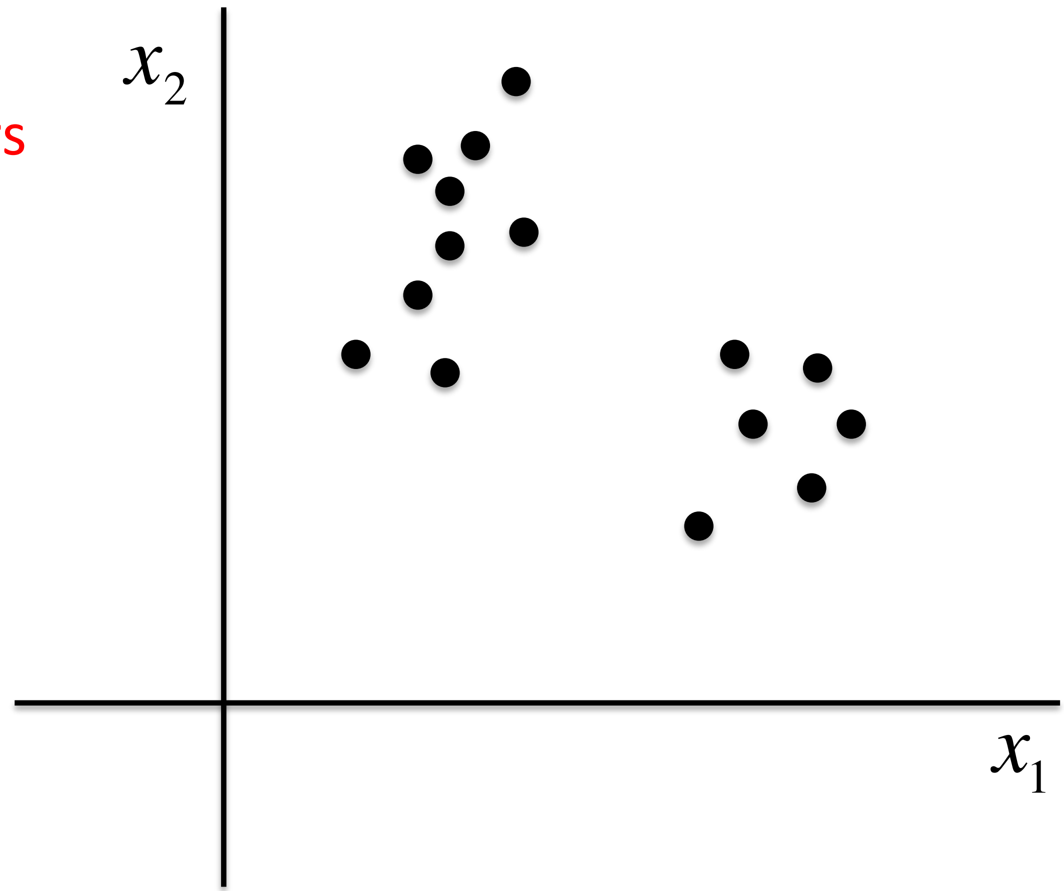
Sample

Features or descriptors

$$\{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_N\}$$

Dataset

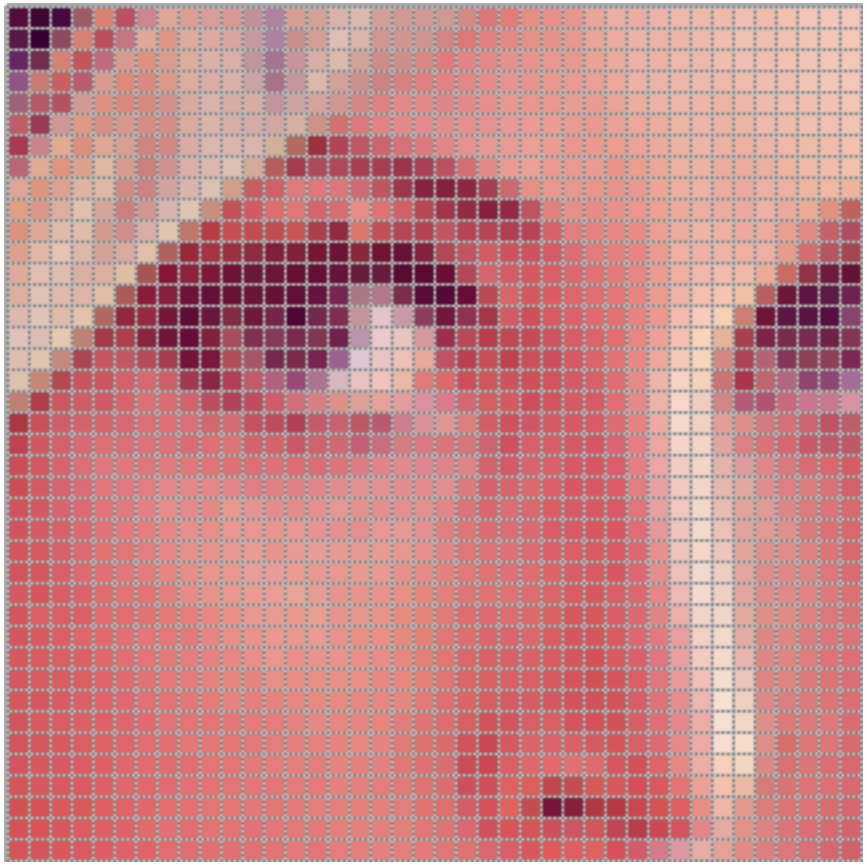
$$\bar{x} = (x_1, x_2)^T$$



Problem Description

$$\overline{x} = (x_1, \ldots, x_m)^T$$

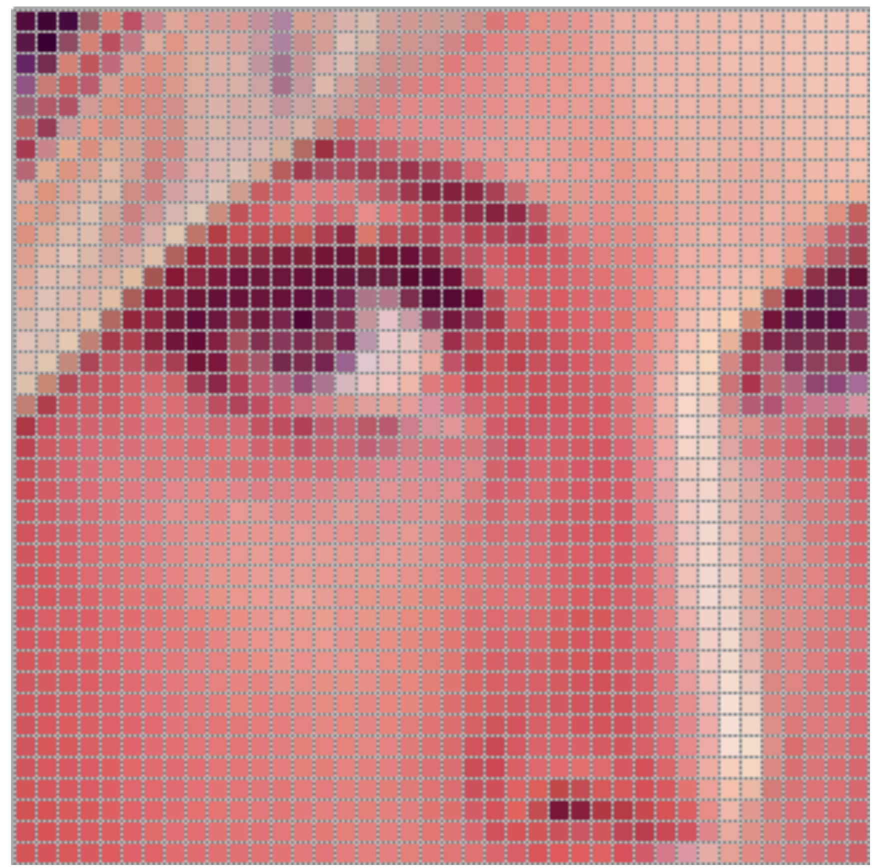
Sample



$$\overline{x} = (x_1, \ldots, x_m)^T$$

Sample

1 Mpx => 10^6 features



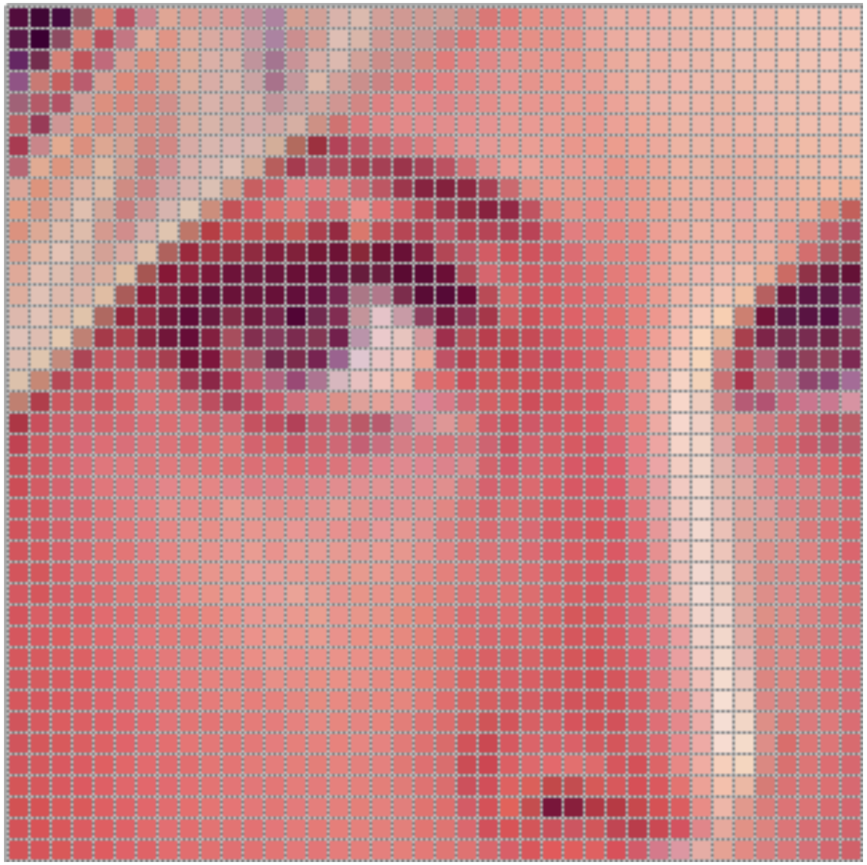
PROBLEM DESCRIPTION

$$\overline{x} = (x_1, \ldots, x_m)^T$$

Sample

1 Mpx => 10^6 features

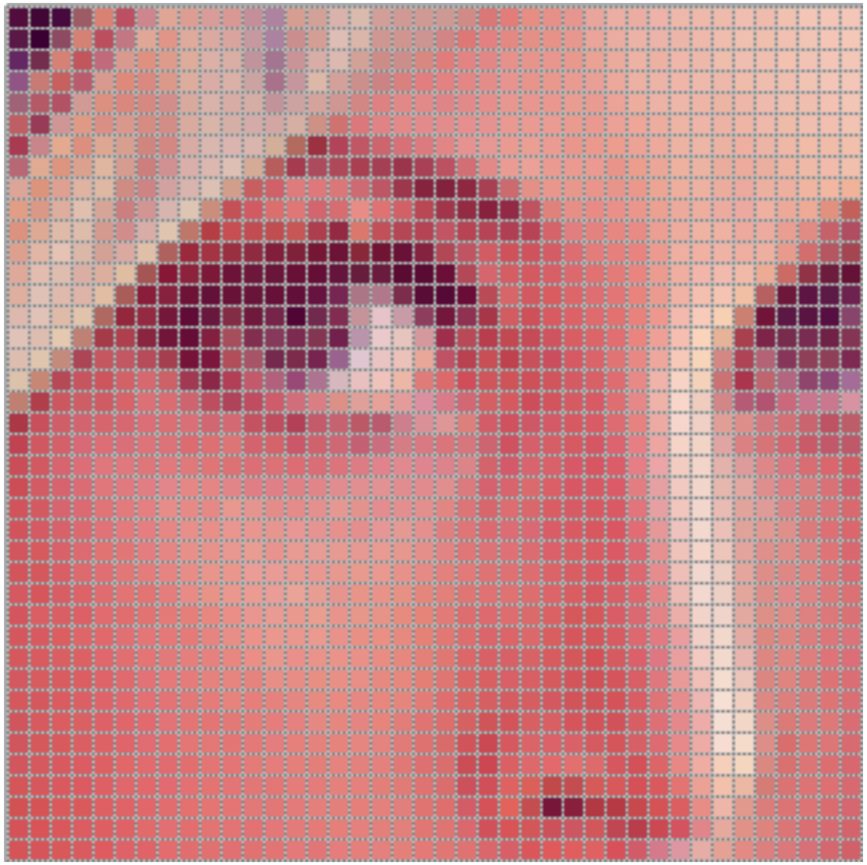
- Computationally expensive
- Curse of dimensionality



$$\overline{x} = (x_1, \ldots, x_m)^T$$

Sample

- Mean, variance, entropy,…
- SIFT
- Harris Corner
- ….

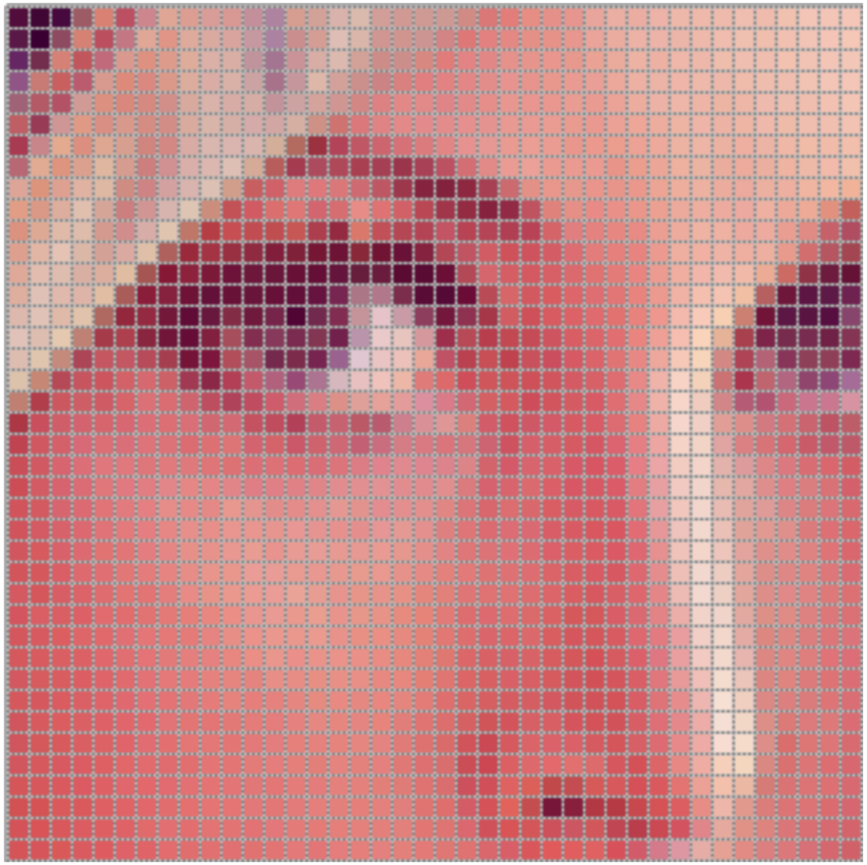


PROBLEM DESCRIPTION

$$\overline{x} = (x_1, \ldots, x_m)^T$$

Sample

Feature extraction is essentially a dimensionality reduction problem with the goal of finding meaningful projections of the original data vectors



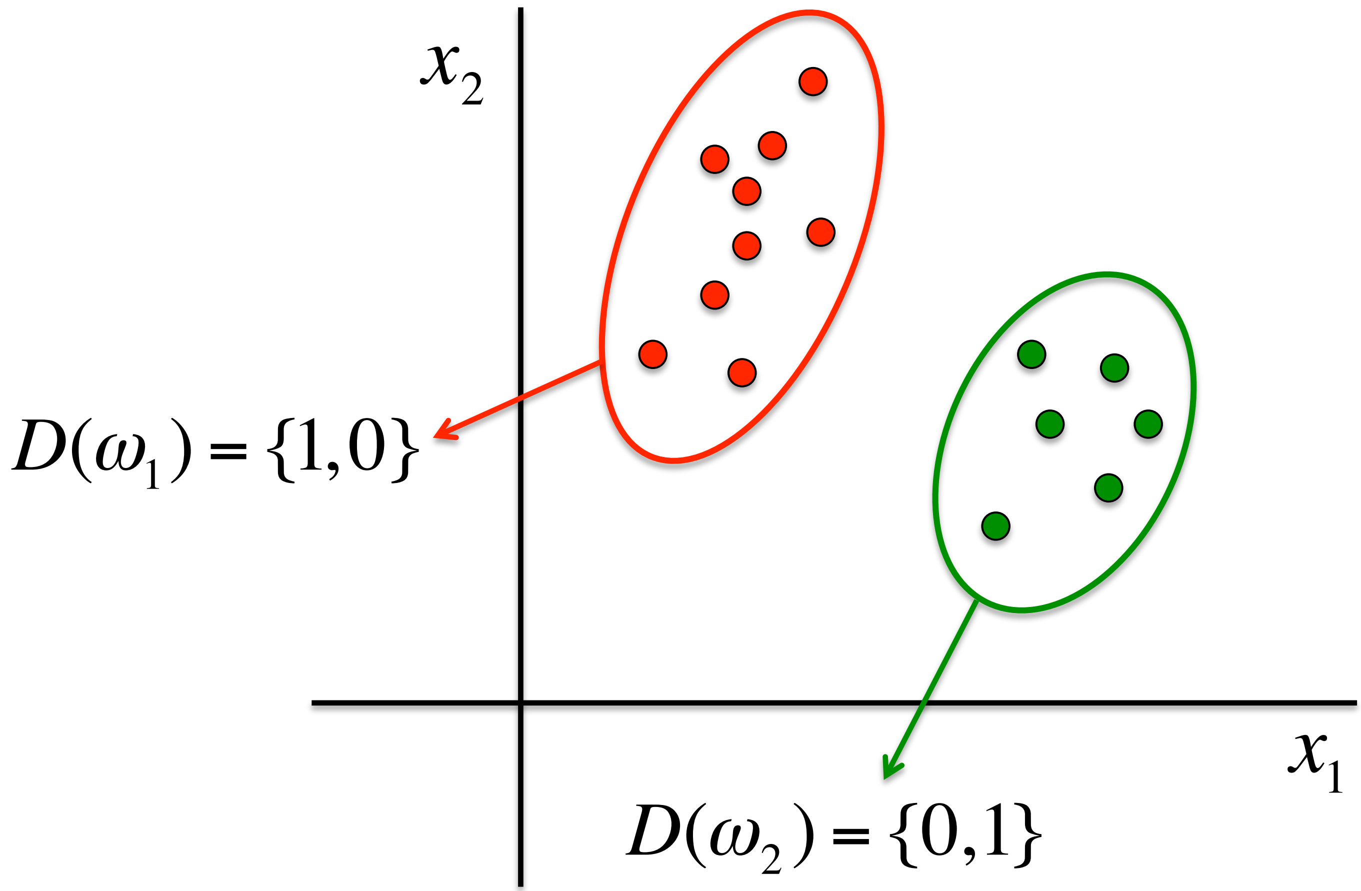
CLASSIFICATION PROBLEM

$$\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$$

Class

$$D(\omega_i) = \{0, 0, \ldots, 1, \ldots, 0\}$$

Label



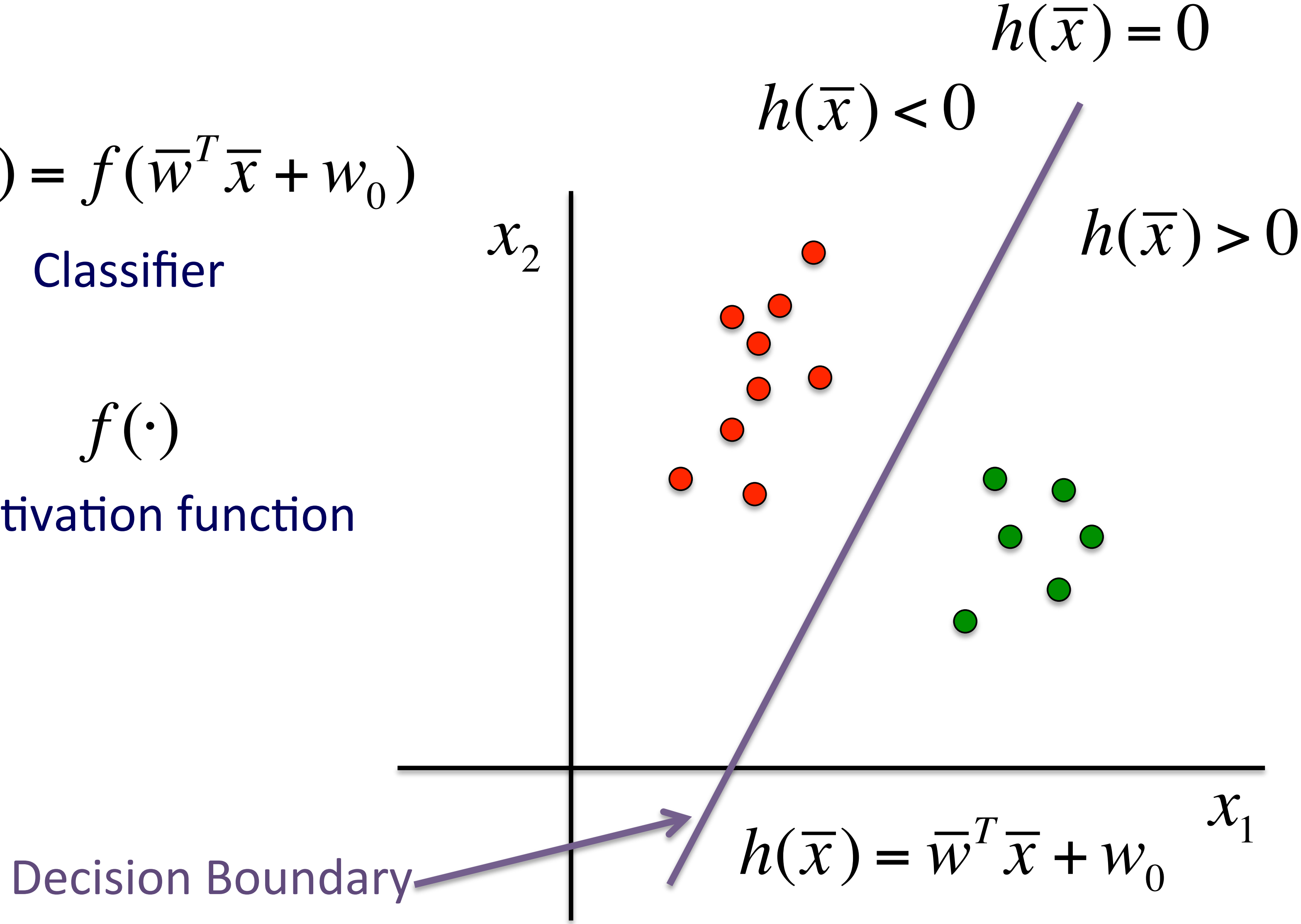
Classification Problem

$$h(\overline{x}) = f(\overline{w}^T \overline{x} + w_0)$$

Classifier

$$f(\cdot)$$

Activation function



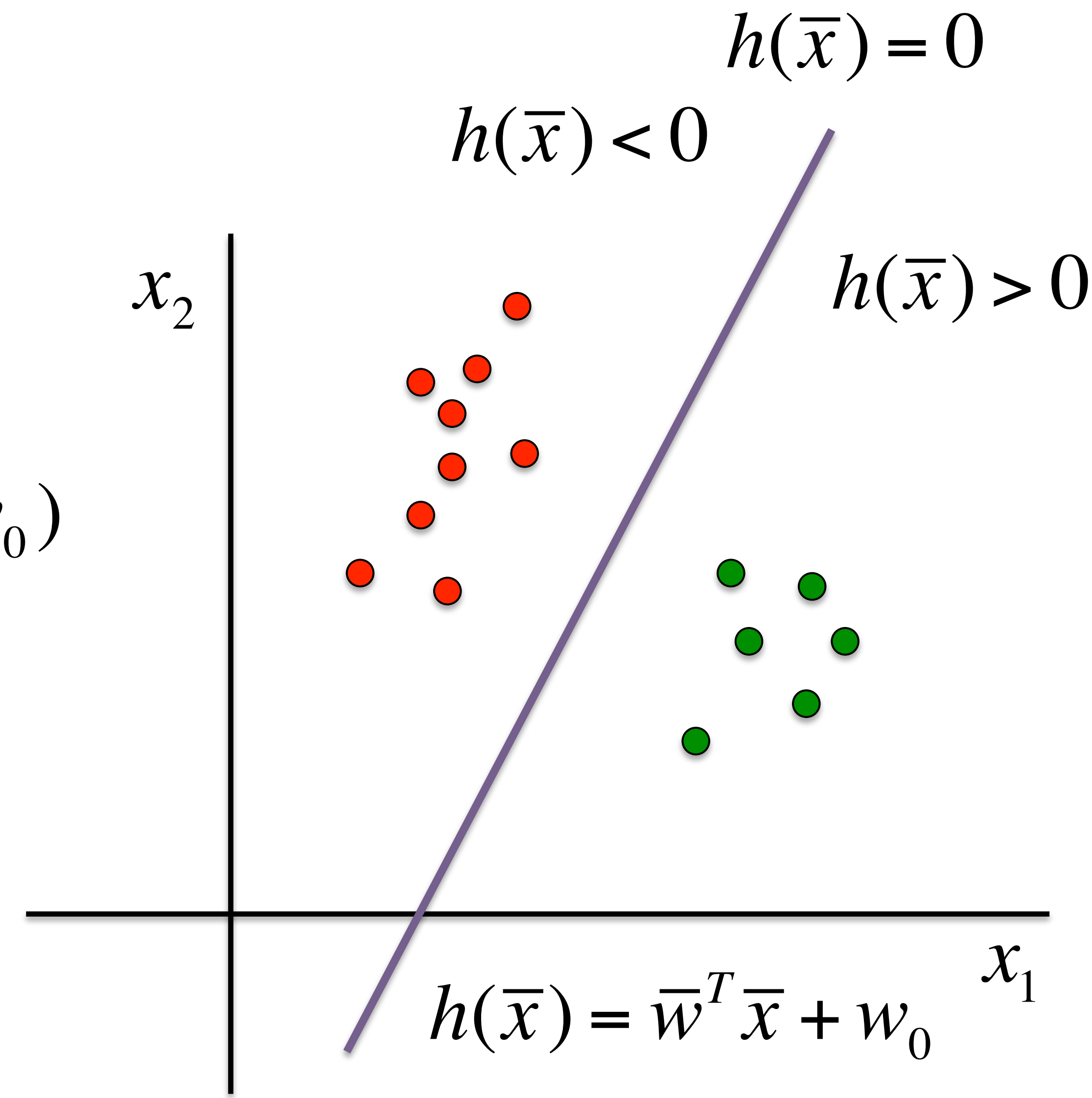
Classifier Training

$$h(\overline{x}) = f(\overline{w}^T \overline{x} + w_0)$$

↑

$$L(\omega(\overline{x}), h(\overline{x}))$$

Loss Function



CLASSIFIER TESTING

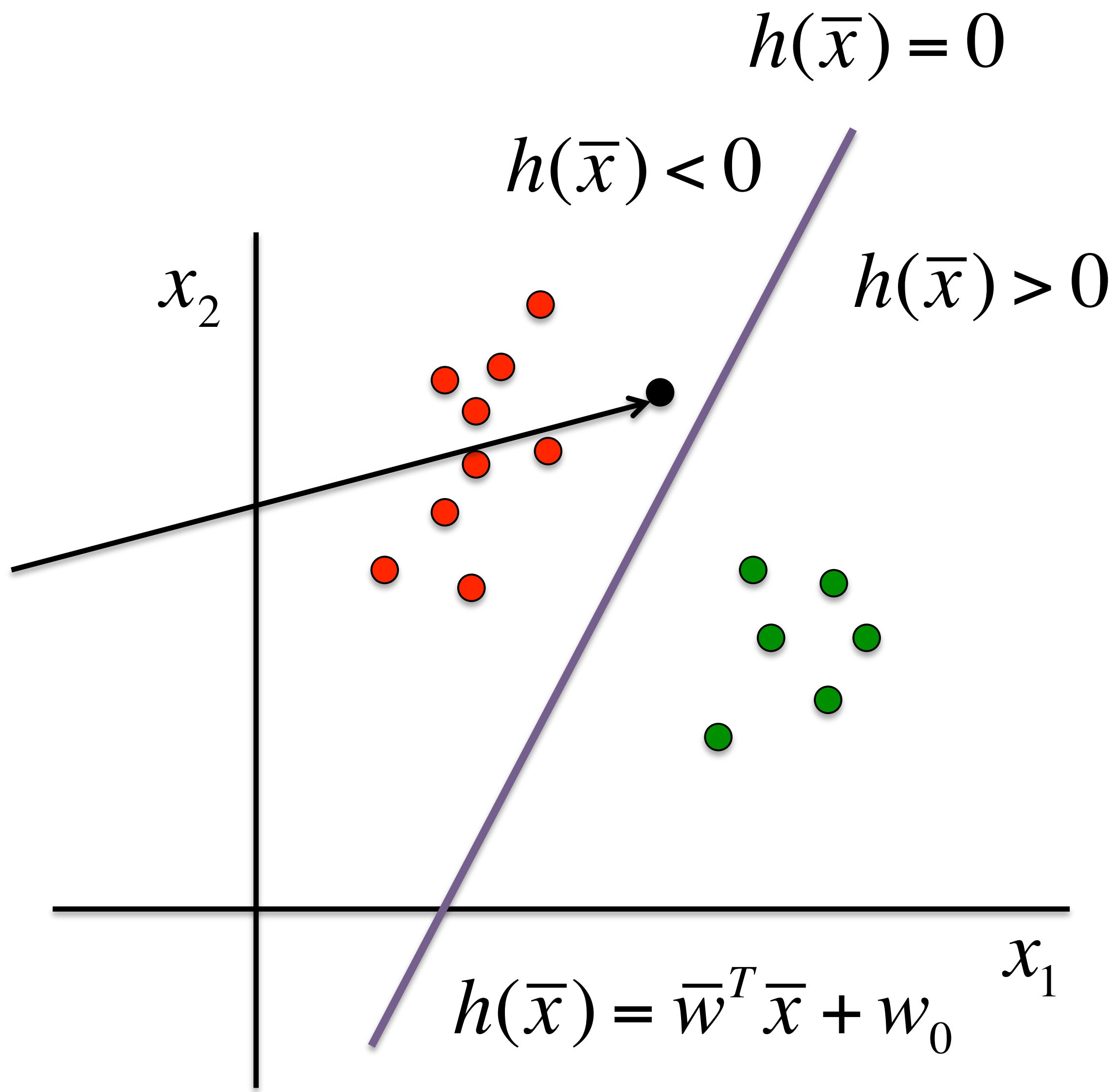
New Sample

$$h(\overline{x}) < 0$$

↓

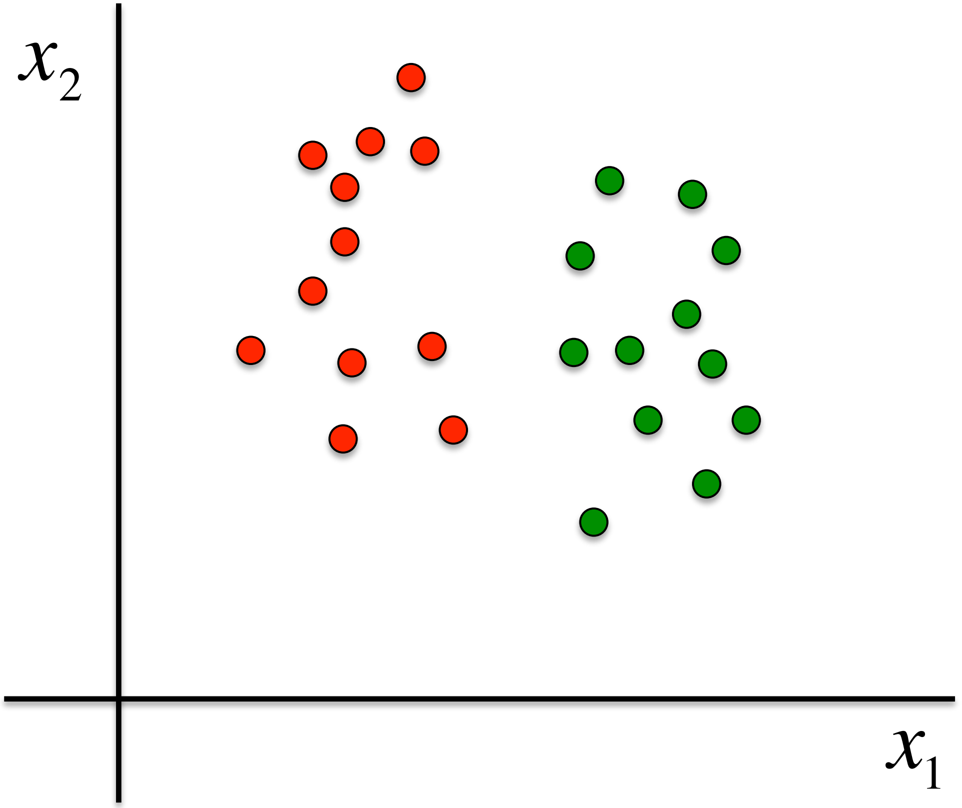
$$D(\omega_1) = \{1,0\}$$

Predicted Class



- k Nearest Neighbors (kNN)
- Neural Network (NN)
- Support Vector Machine

k Nearest Neighbors

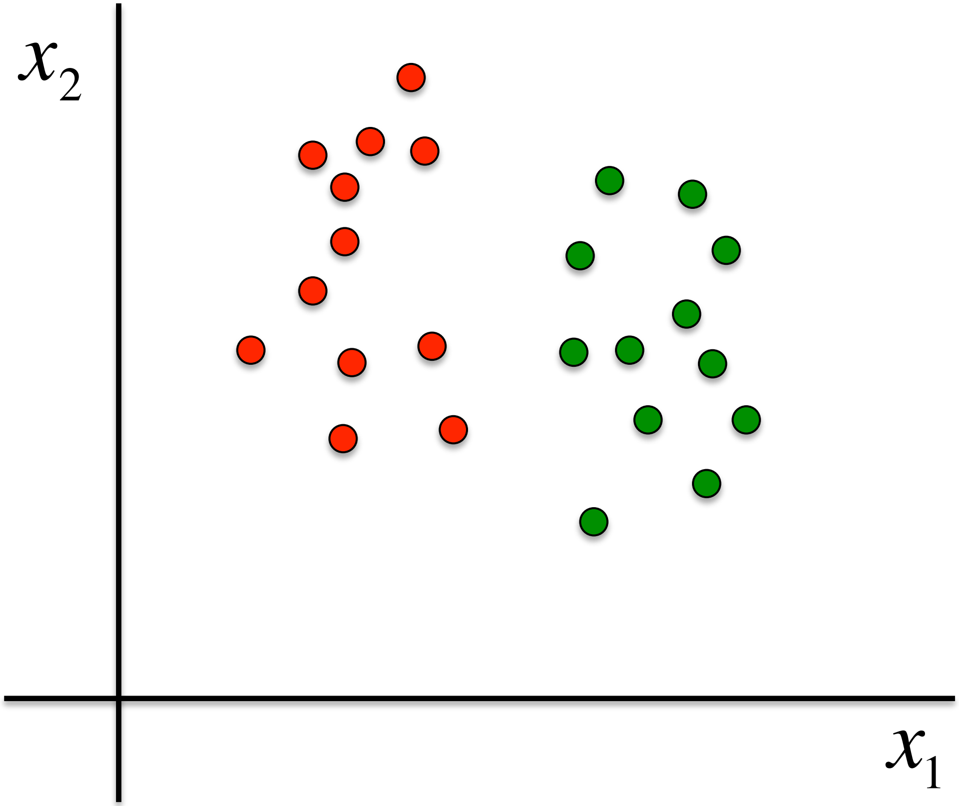


k Nearest Neighbors

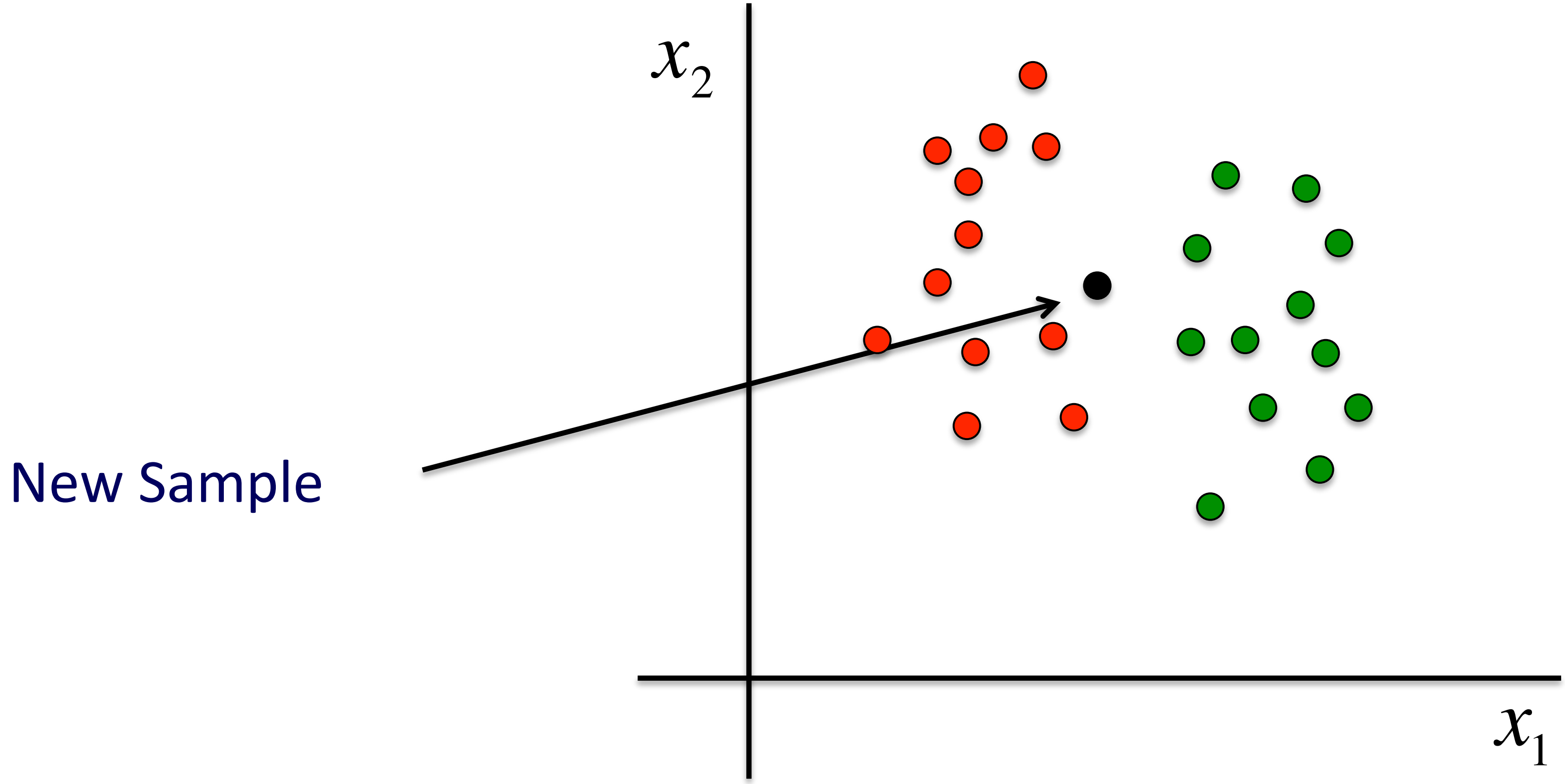
No training set

↓

Reference Set



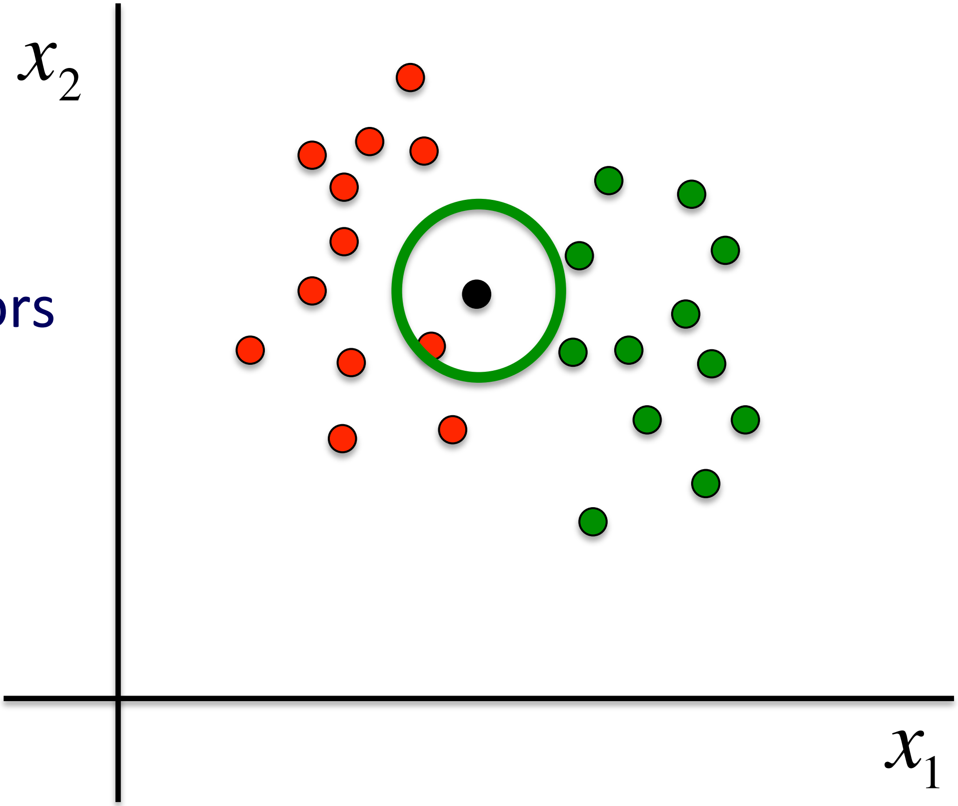
k Nearest Neighbors



k Nearest Neighbors

Algorithm steps:

- Consider the fixed value k
  e.g. K=1
- Find the K nearest neighbors
- Set the class according to a majority voting



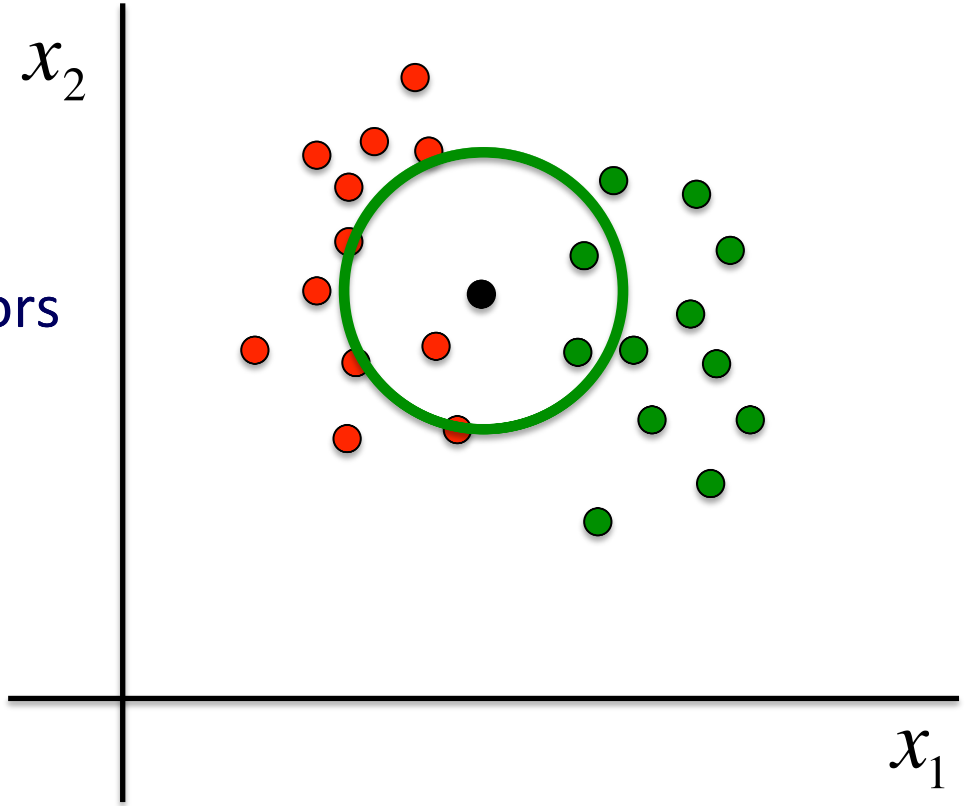
k Nearest Neighbors

- Is one of the simplest machine learning algorithm;
- There is no training stage;
- The classification function is approximated locally;
- Several variants of the original algorithm have been proposed.

k Nearest Neighbors

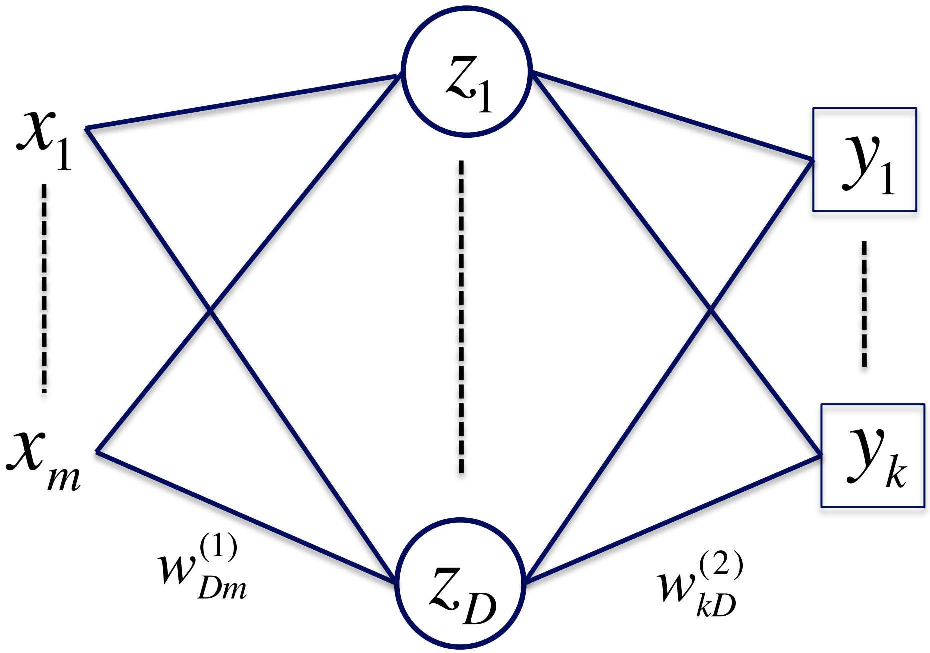
Algorithm steps:

- Consider the fixed value k
  e.g. K=3
- Find the K nearest neighbors
- Set the class according to a majority voting

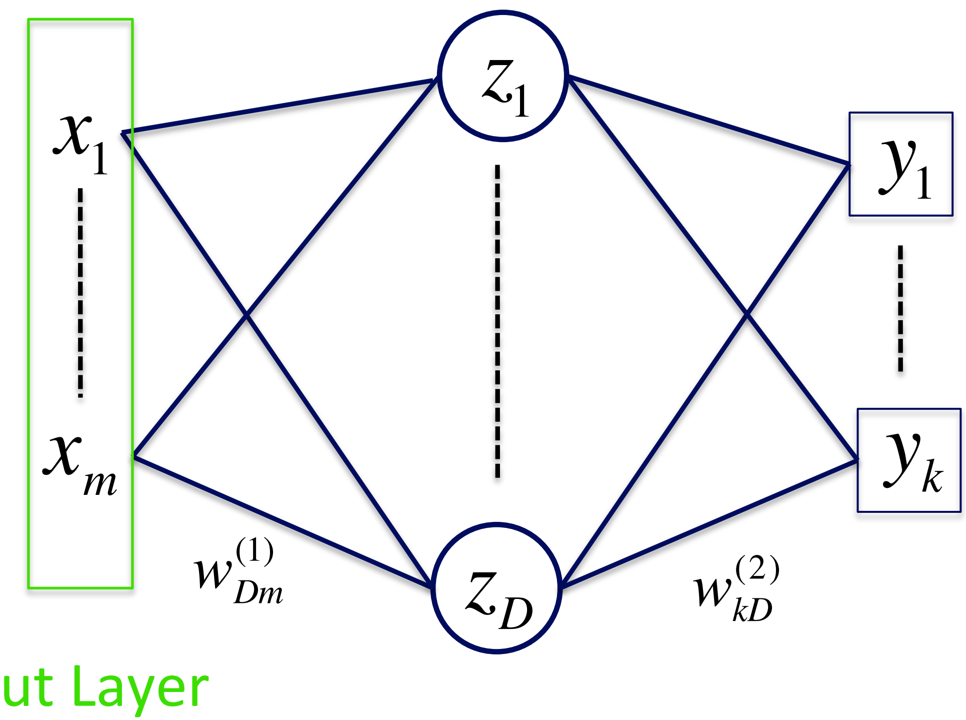


NEURAL NETWORK

Inspired by the central nervous system, a Neural Network is a set of interconnected "neurons" which compute value from inputs

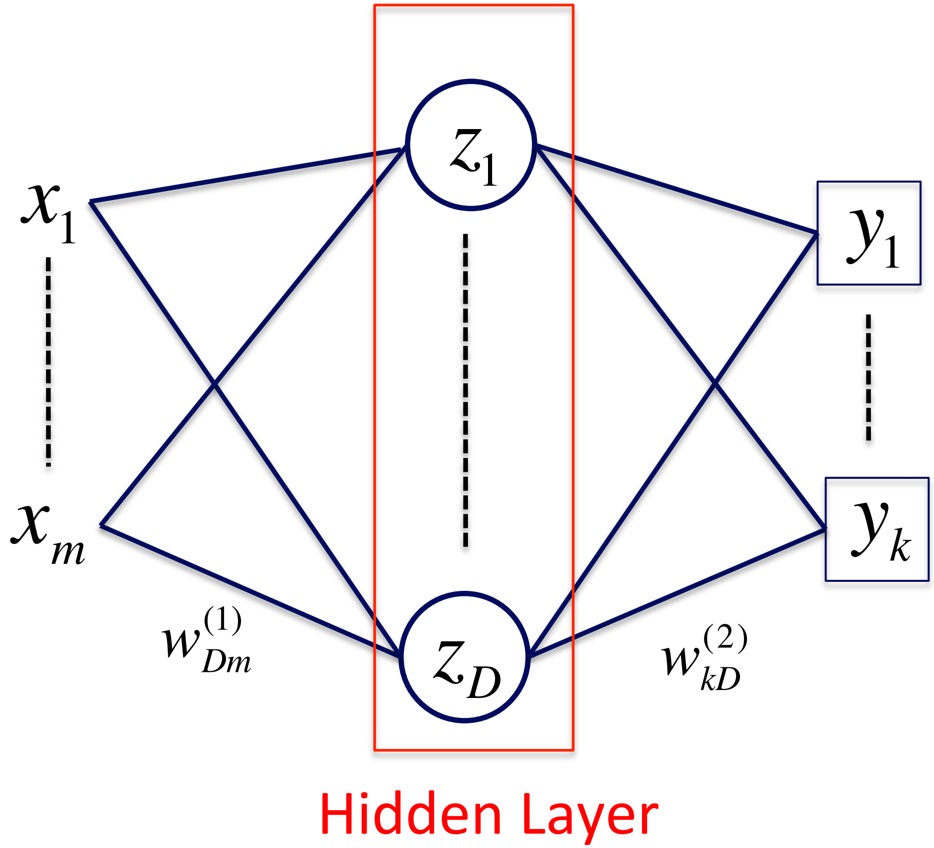


Inspired by the central nervous system, a Neural Network is a set of interconnected "neurons" which compute value from inputs



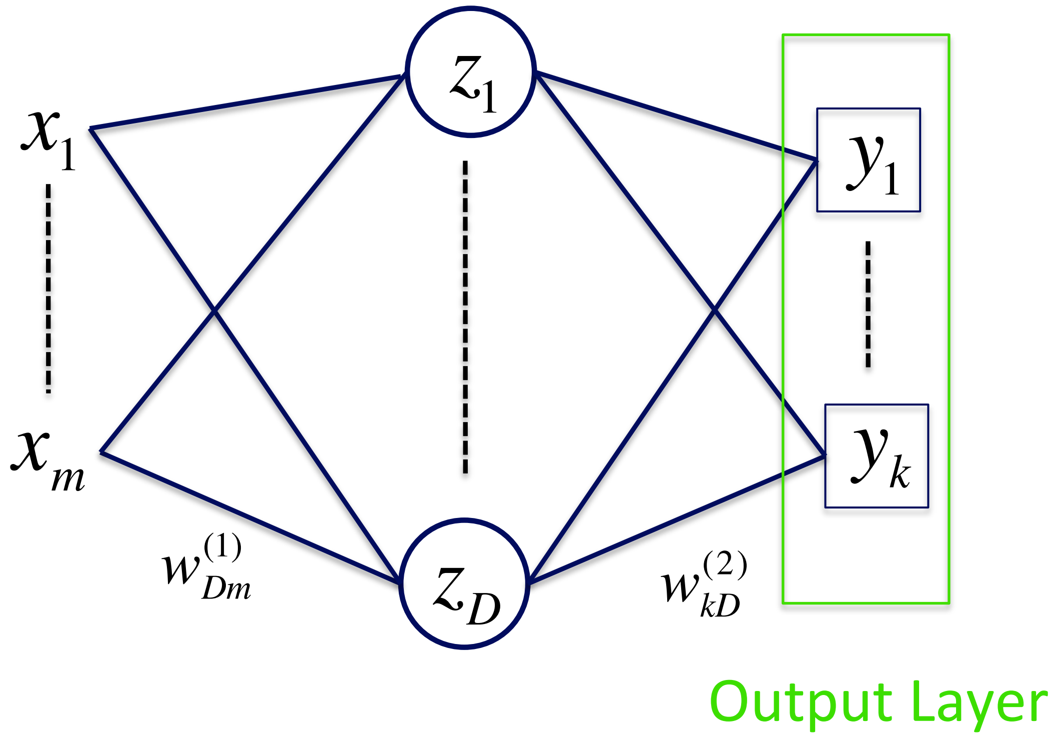
Neural Network

Inspired by the central nervous system, a Neural Network is a set of interconnected "neurons" which compute value from inputs

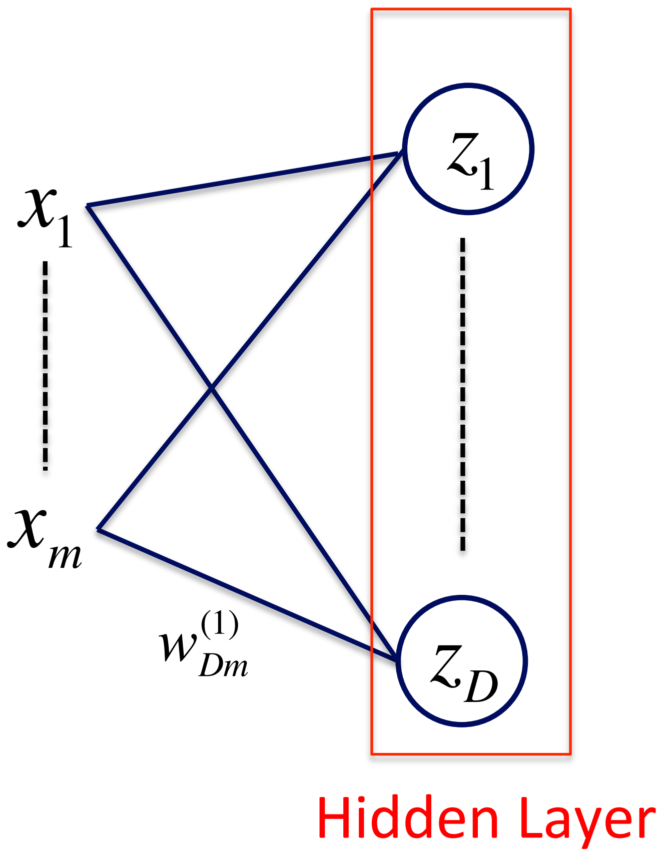


Neural Network

Inspired by the central nervous system, a Neural Network is a set of interconnected “neurons” which compute value from inputs



The value of each hidden unit is computed by an activation function according to the value of weights of the first layer and the inputs.



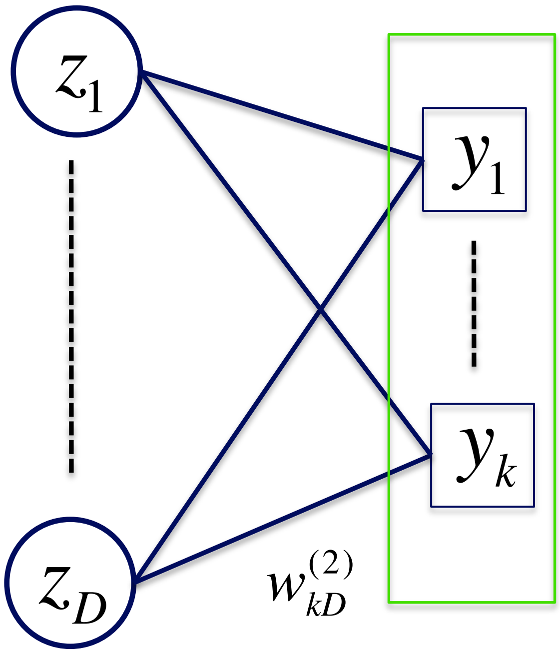
$$z_j = h(a_j)$$

Activation function

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i$$

Input Linear Combination

The output value is computed by an activation function according to the values of the weigths of the second layer and the hidden units.



Output Layer

$$y_k = \sigma(a_k)$$

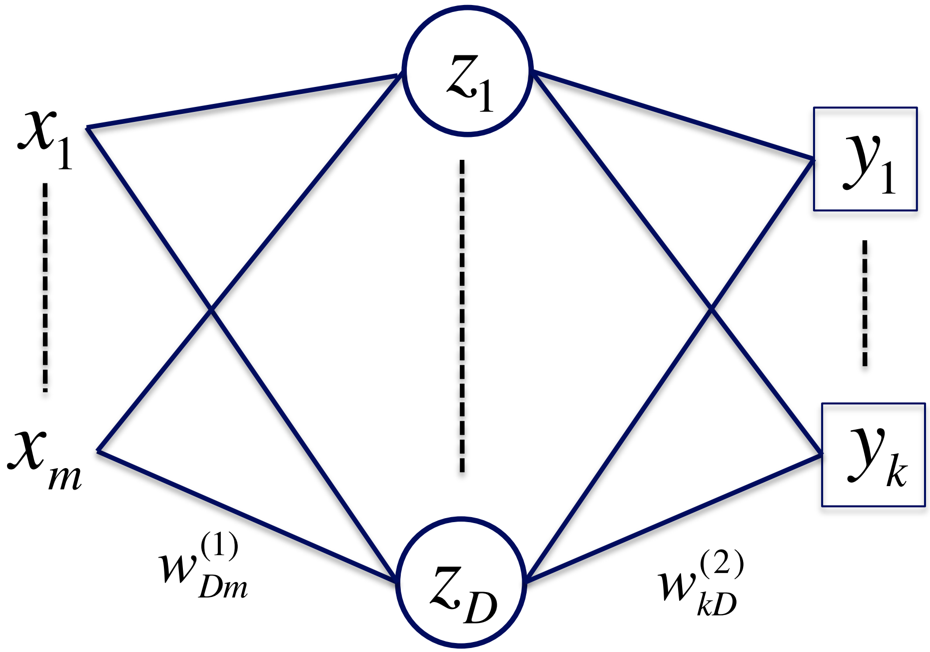
Activation function

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j$$

Output unit activations

NEURAL NETWORK

The training of the network consists in estimating the layers' weights



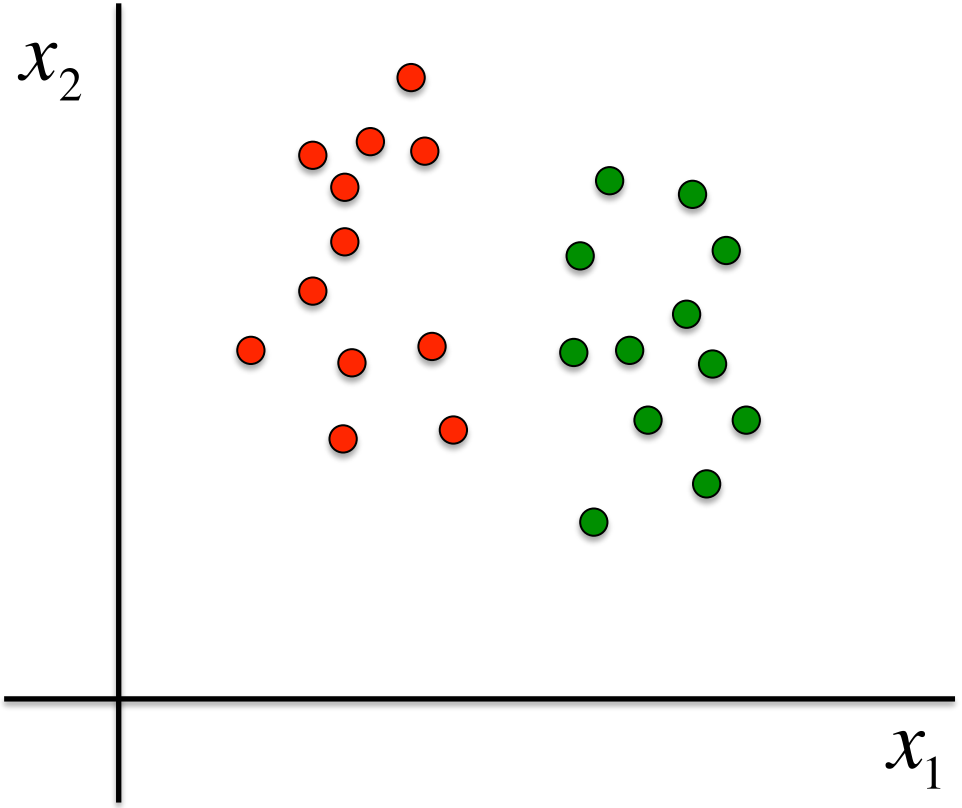
$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\left\| y(\bar{x}, w) - D(\omega)\right\|^2$$

- Activation function generally are sigmoid function

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$

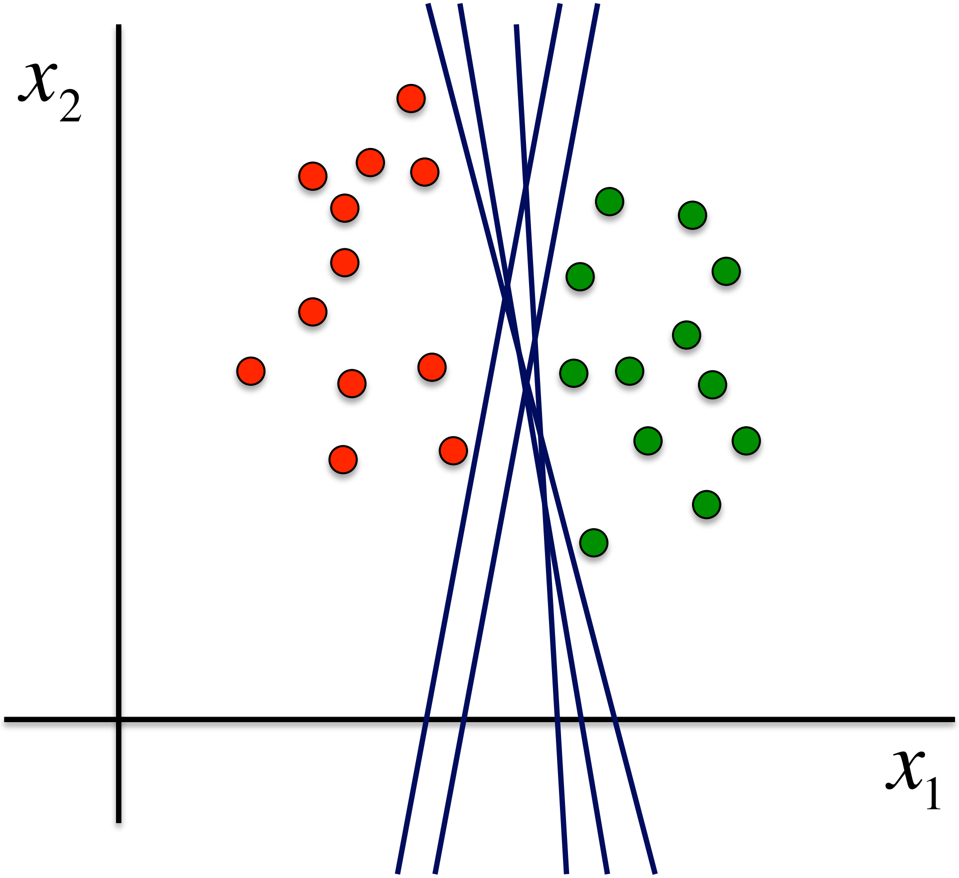
- The number of layers and the number of the hidden units are hyper-parameters of the model

In the support vector machines the decision boundary is chosen to be the one for which the margin is maximized.



In the support vector machines the decision boundary is chosen to be the one for which the margin is maximized.

$$y(\overline{x}_n) = \overline{w}^T \phi(\overline{x})$$

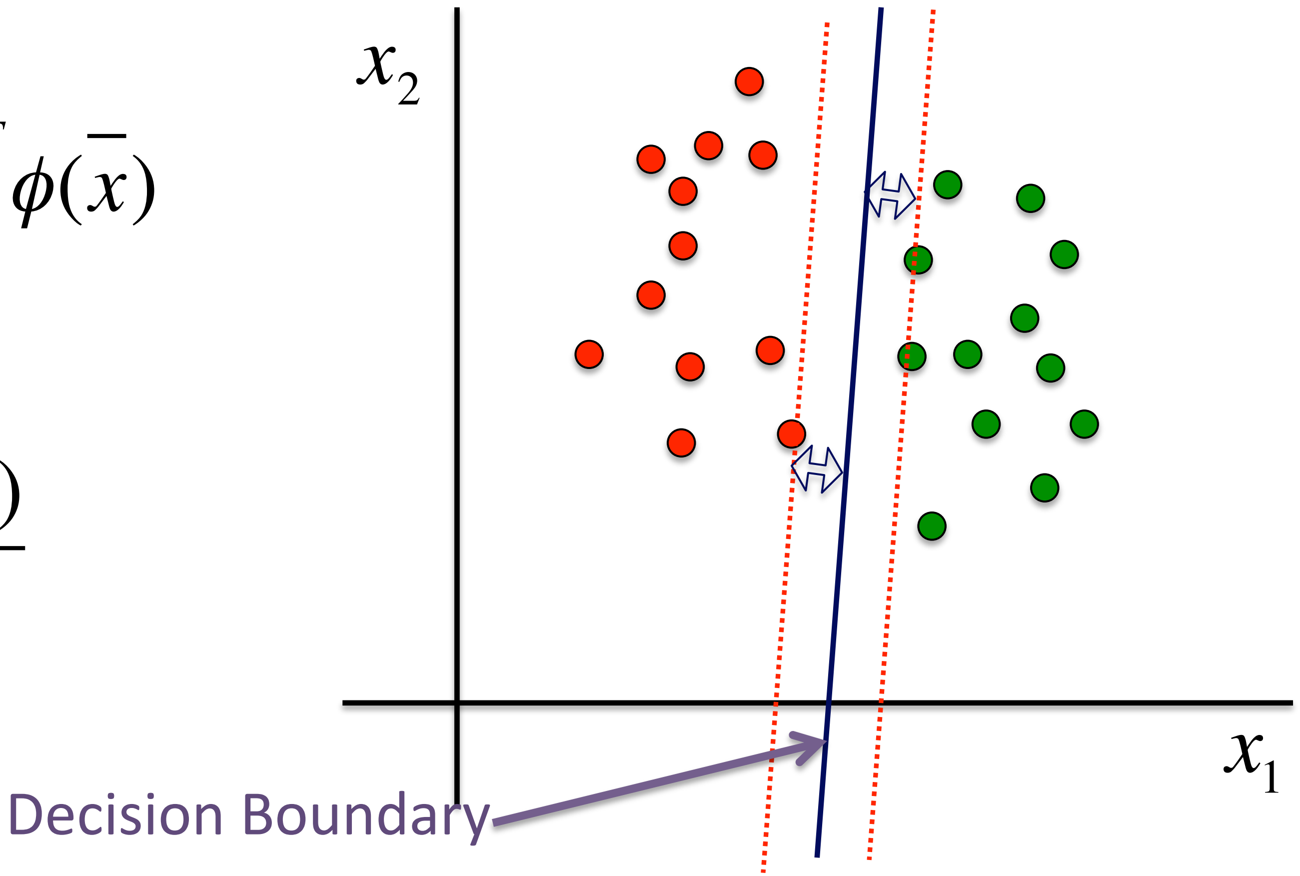


SUPPORT VECTOR MACHINE

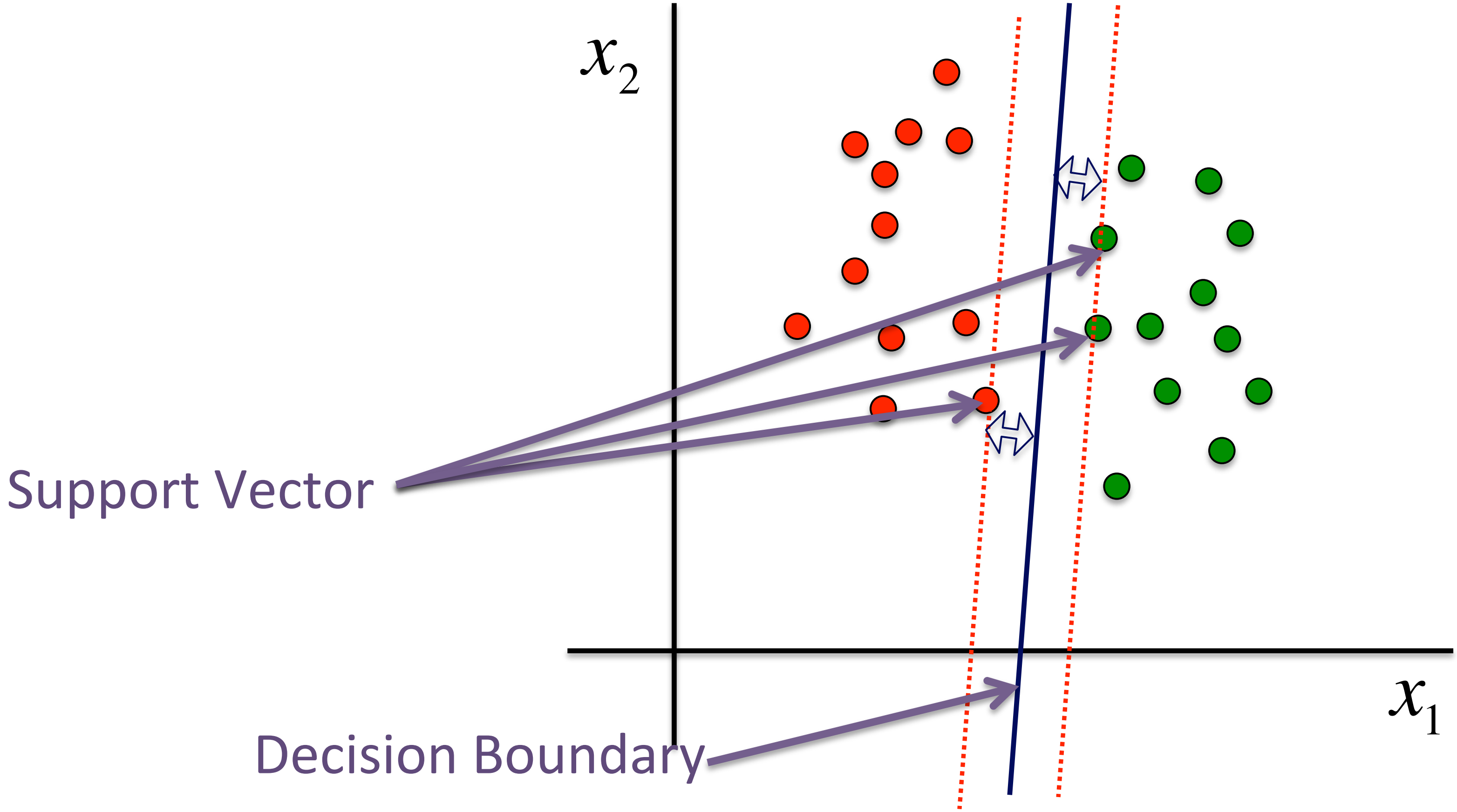
In the support vector machines the decision boundary is chosen to be the one for which the margin is maximized.

$$y(\bar{x}_n) = \bar{w}^T \phi(\bar{x})$$

$$\frac{t_n y(\bar{x}_n)}{\|\bar{w}\|}$$



In the support vector machines the decision boundary is chosen to be the one for which the margin is maximized.



An important property of support vector machines is that the determination of the model parameters corresponds to a convex optimization problem, and so any local solution is also a global optimum.

$$L(w,a) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N} a_{\mathrm{n}} \left\{ t_n \overline{w}\phi(x_n) + b) - 1 \right\}$$

$$y(\overline{x}) = \sum_{n=1}^{N} a_n t_n k(\overline{x}, \overline{x}_n)$$

An important property of support vector machines is that the determination of the model parameters corresponds to a convex optimization problem, and so any local solution is also a global optimum.

Lagrange Multiplier

Ground truth

Loss Function

$$L(w,a) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n \overline{w}\phi(x_n) + b) - 1 \right\}$$

Prediction

Model

$$y(\overline{x}) = \sum_{n=1}^{N} a_n t_n k(\overline{x}, \overline{x}_n)$$

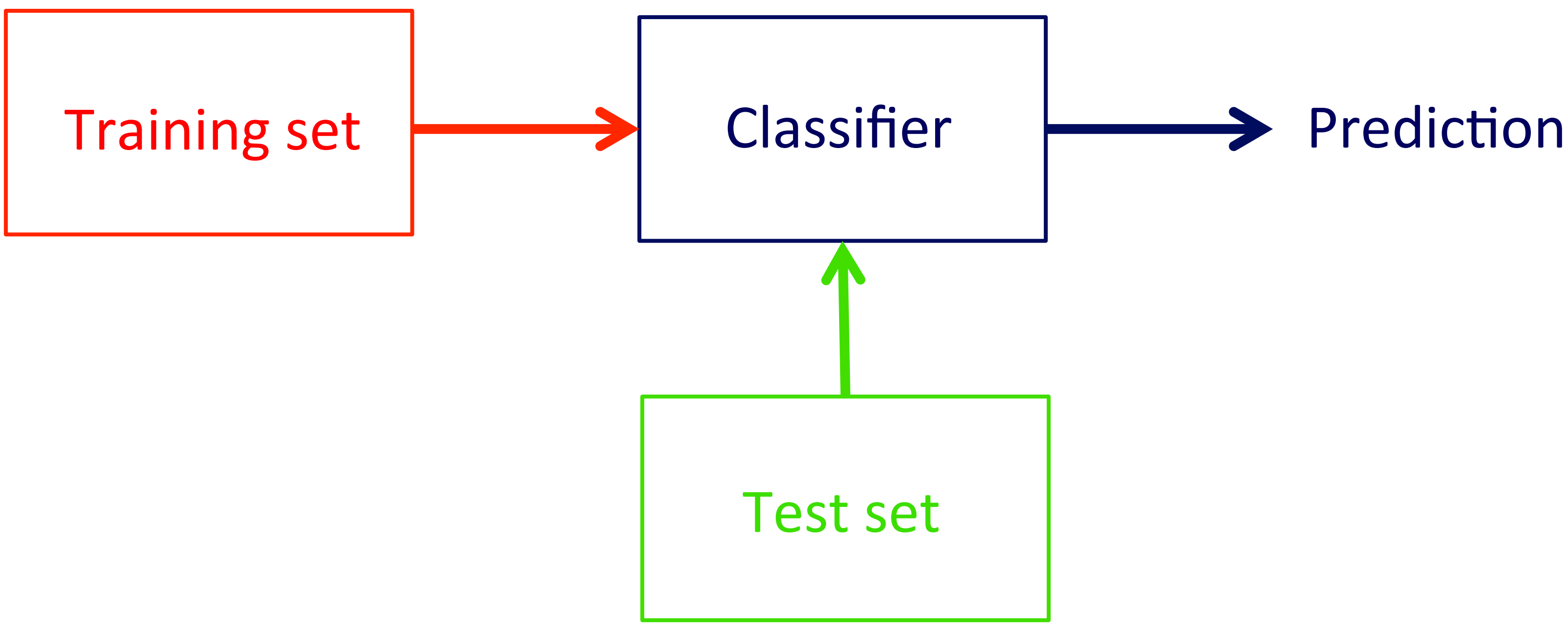
Kernel

- The SVM is widely used is several domains.

- Several variants of the algorithm and kernel types have been proposed in the literature.

- The classifier, according also to the kernel used, requires the tuning of some hyper-parameters.

Classification scheme

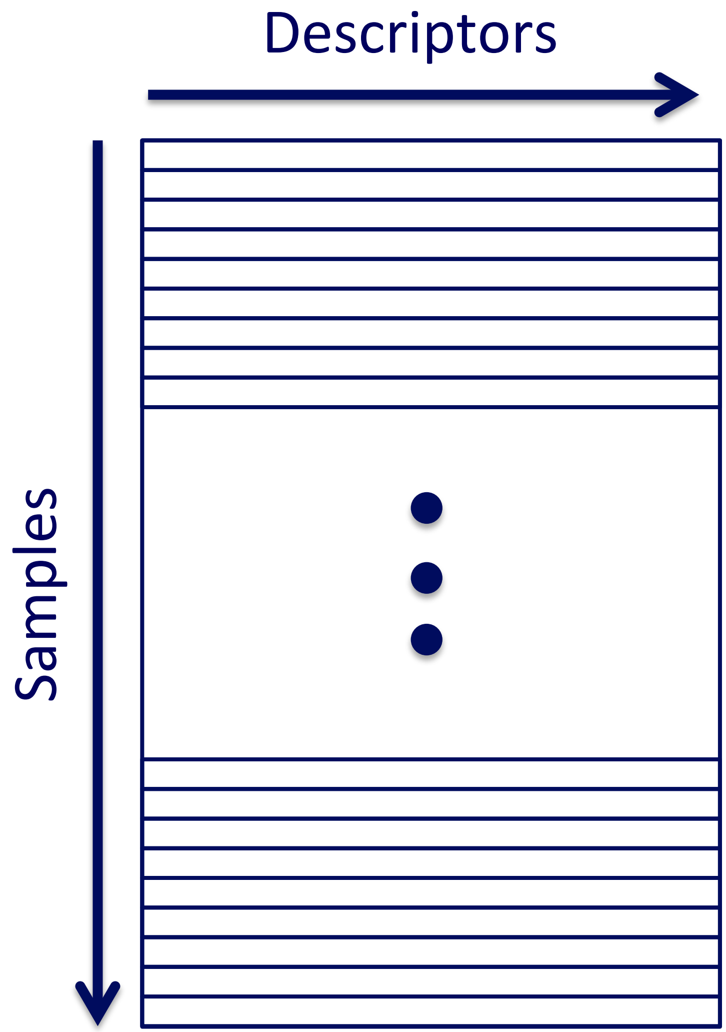


Validation

The classification model depends on the samples used in the training stage. The ability of the system to recognize new samples depend on the parameters used.

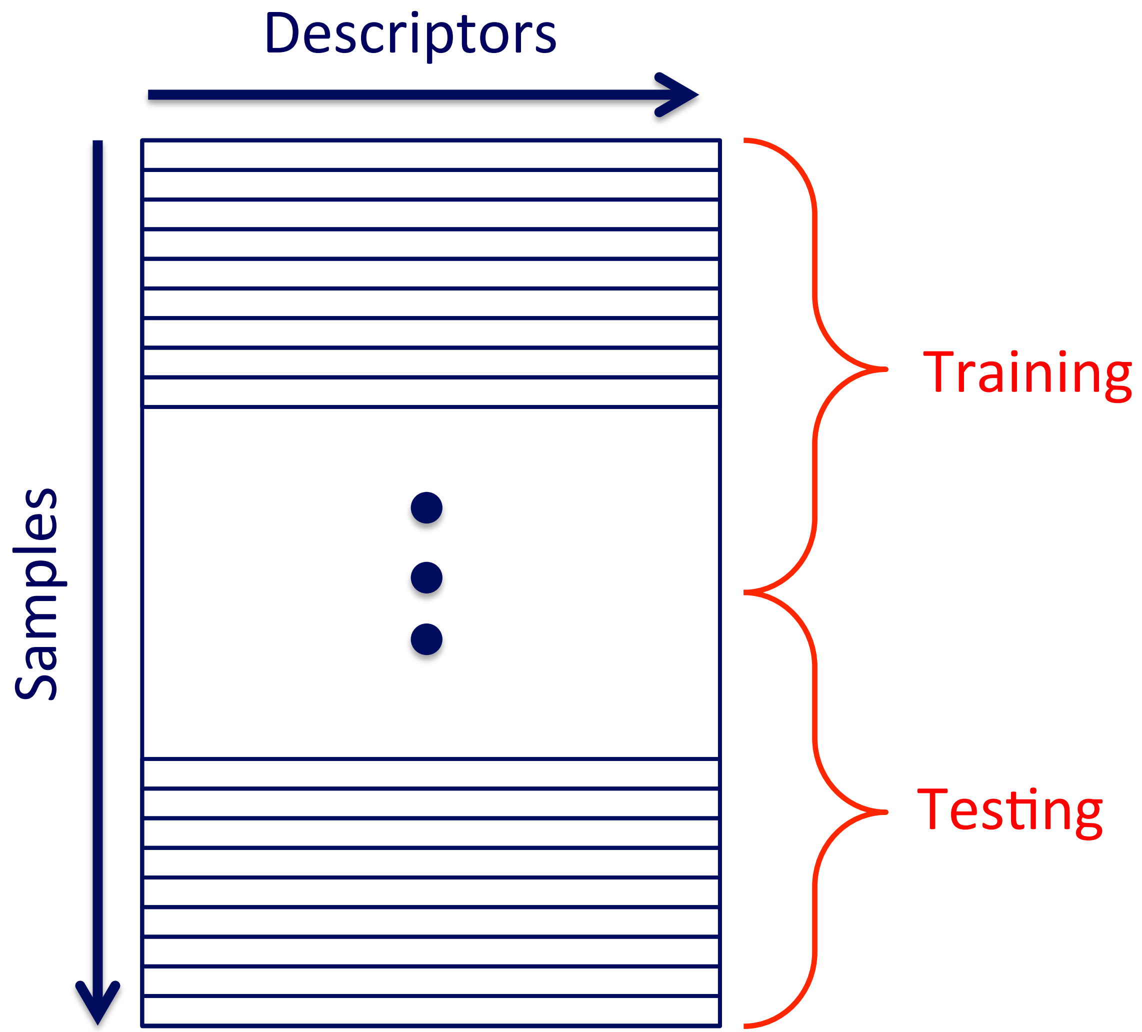
- How to perform tests independently of the samples used
- How to chose the optimal classifier' hyper-parameters

Random Sub-Sampling



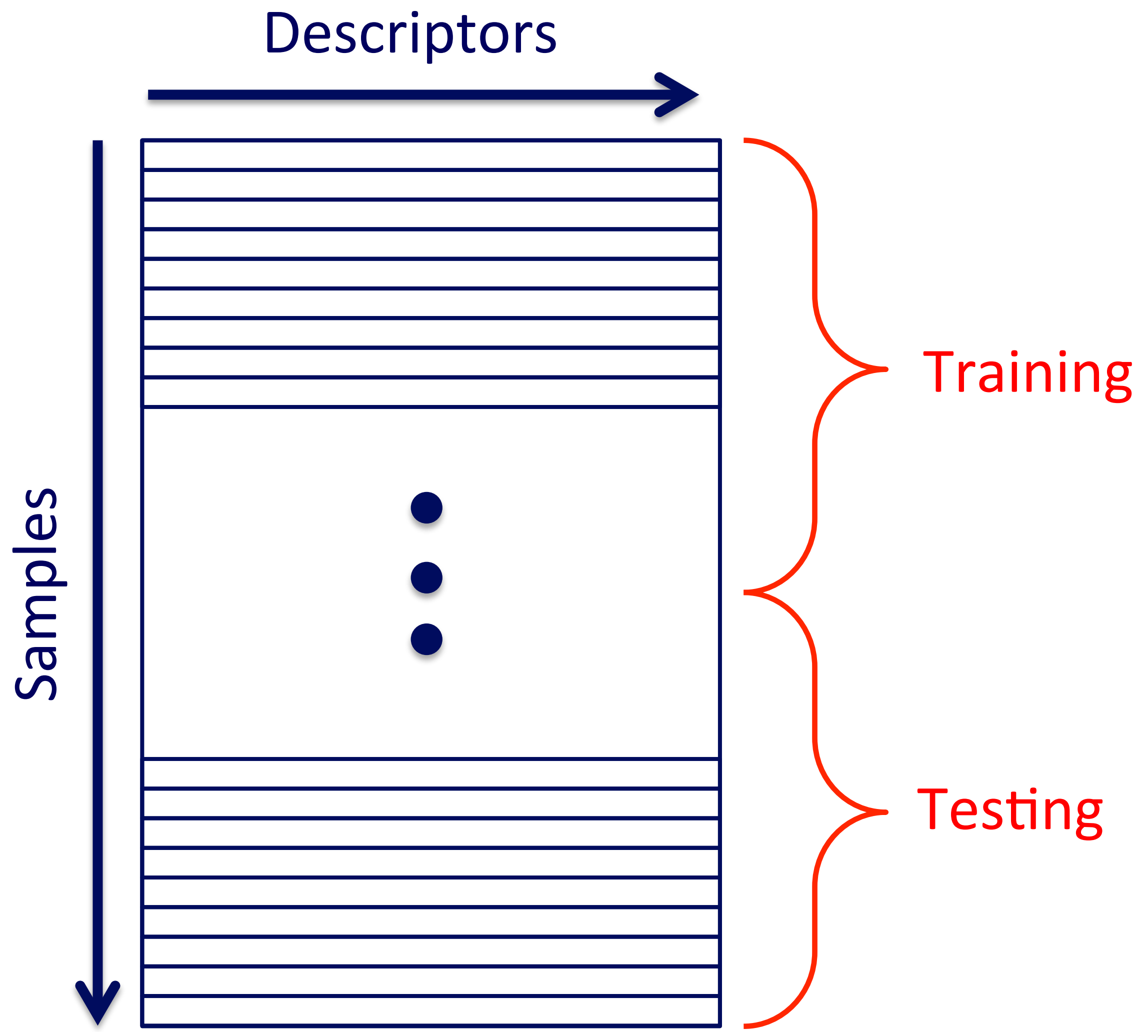
Which is the best way to generate the training set and the test set from the original dataset?

Random Sub-Sampling



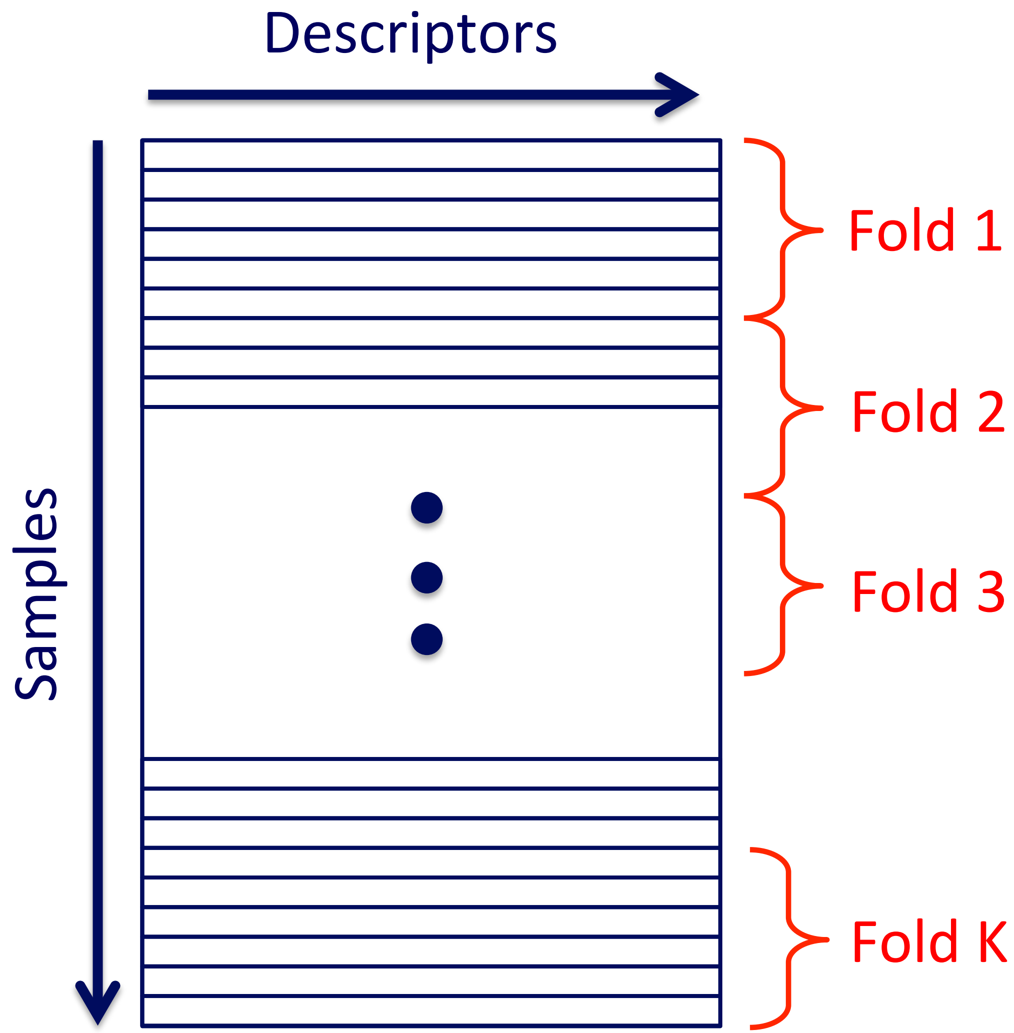
Validation

Random Sub-Sampling



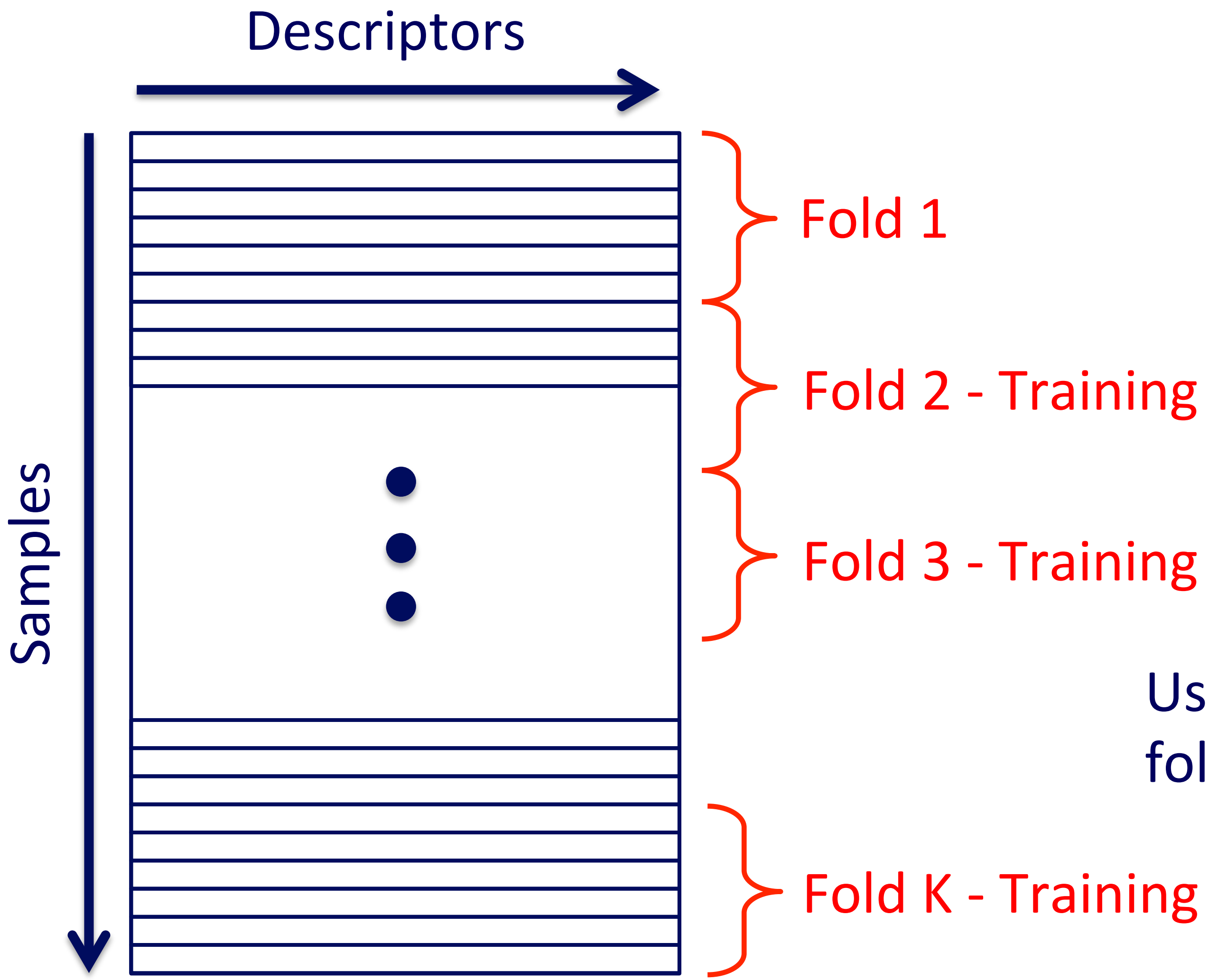
The model depends on the samples used in the training

K FOLD CROS-VALIDATION



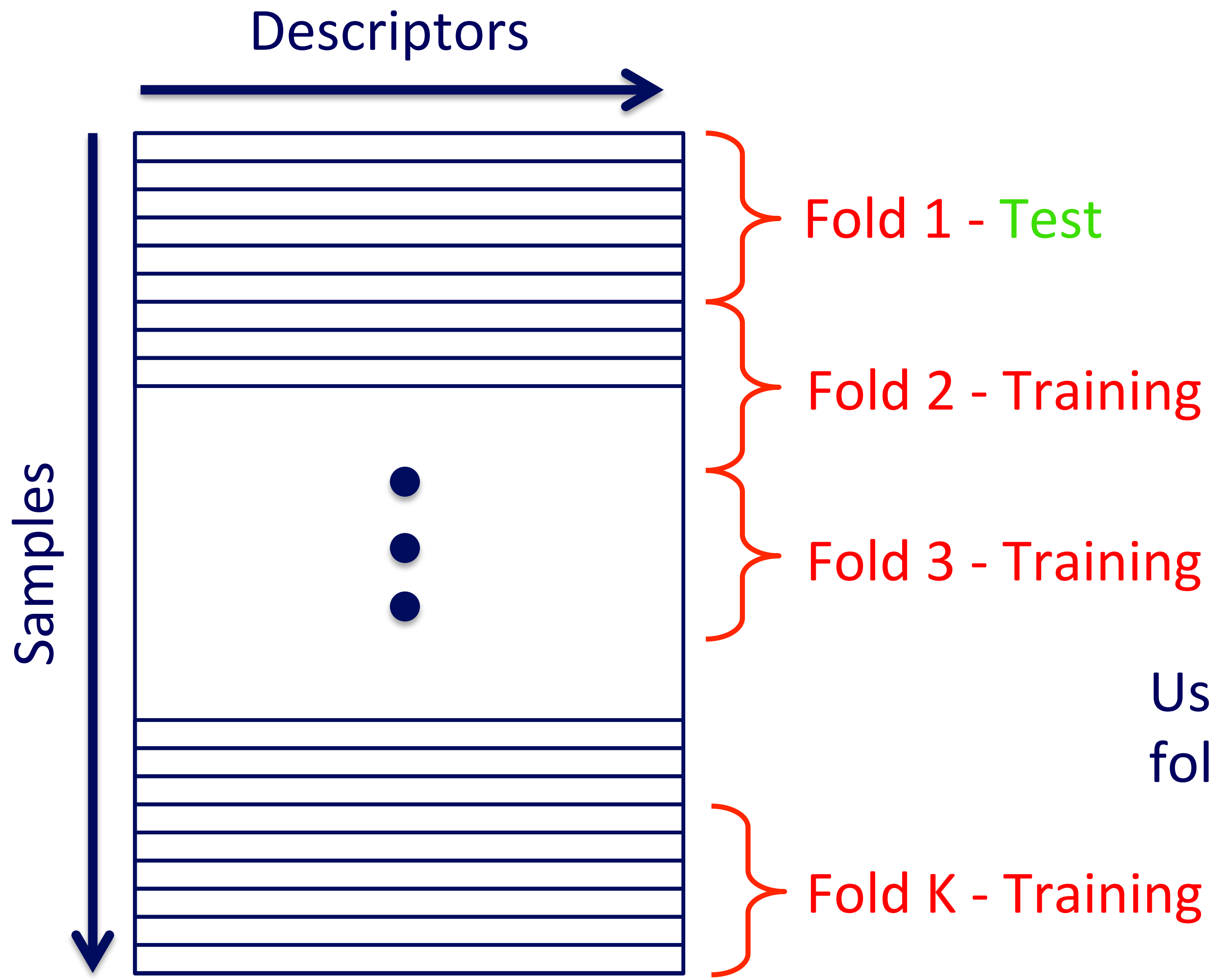
Define k independent folds

K fold Cros-Validation



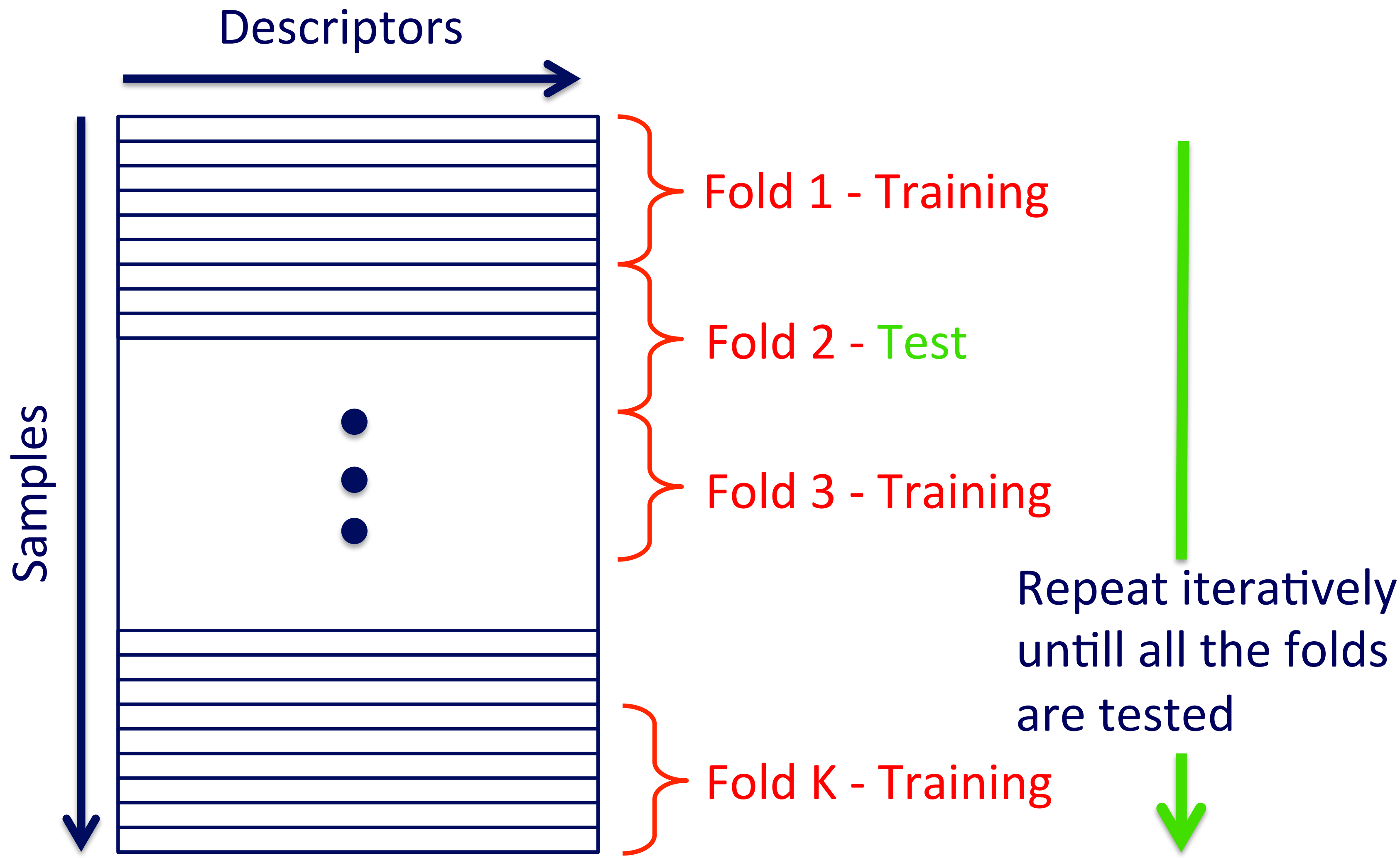
Use k-1
folds for training

K Fold Cros-Validation

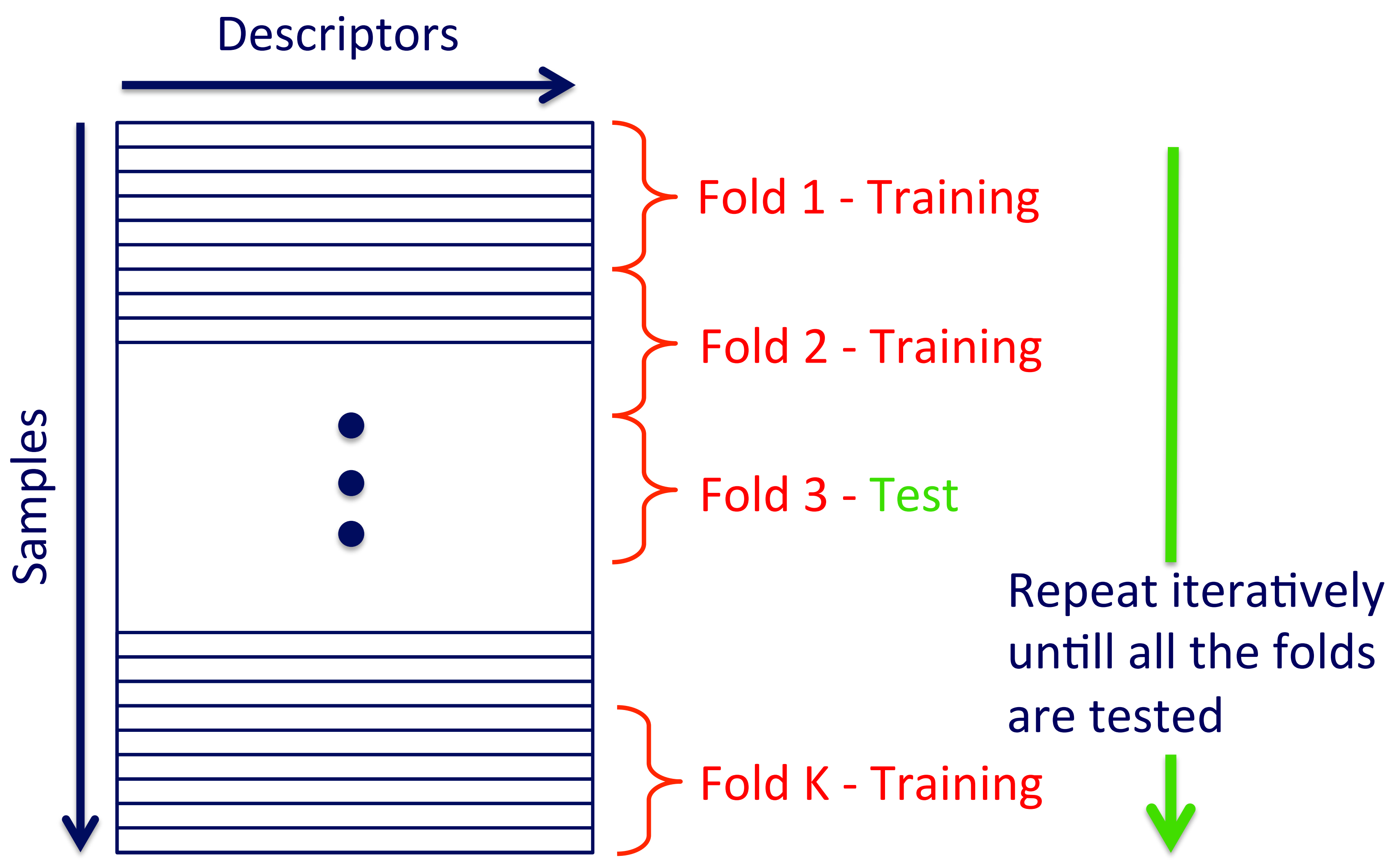


Use the remaining fold for the test

K FOLD CROS-VALIDATION

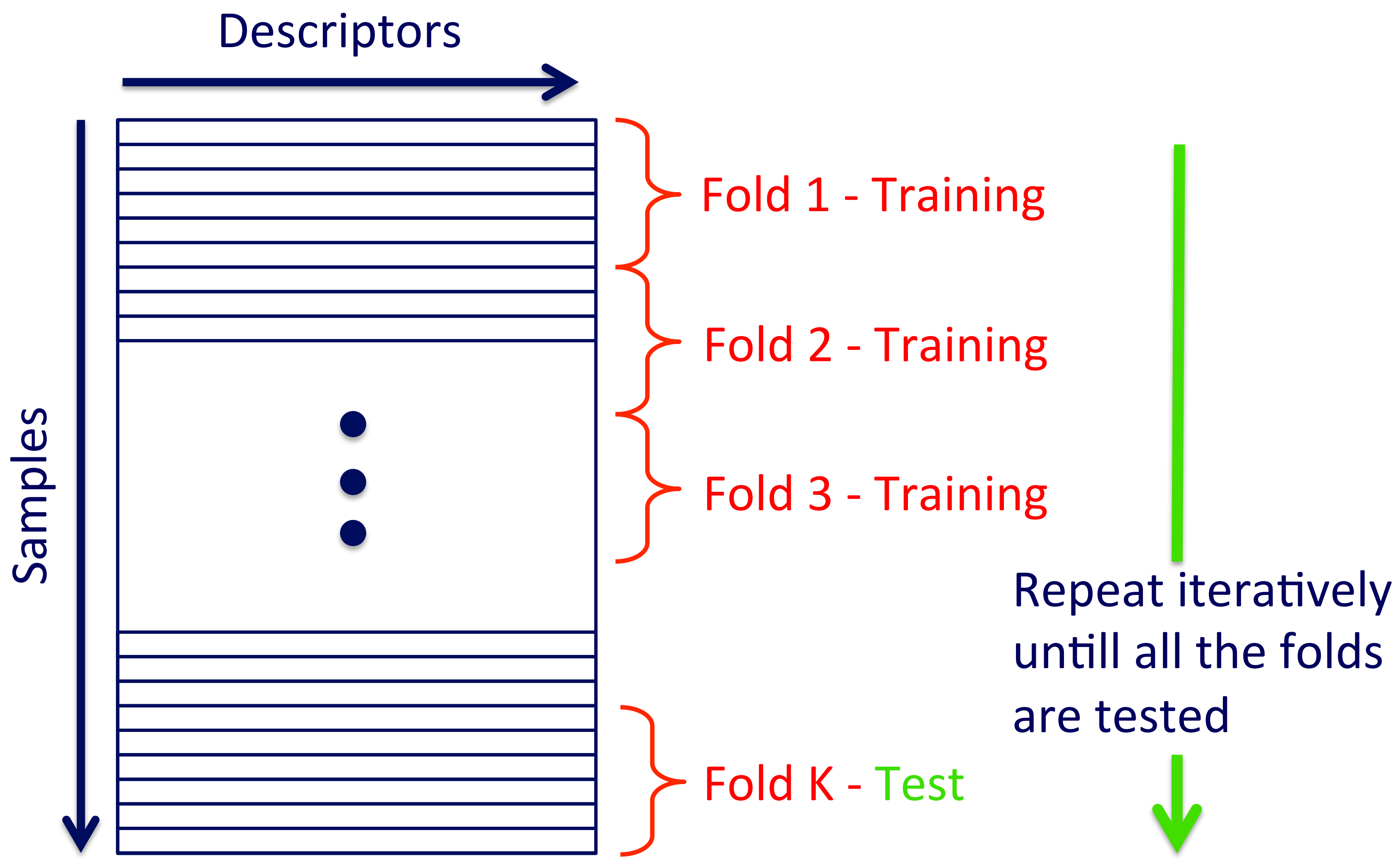


K FOLD CROS-VALIDATION

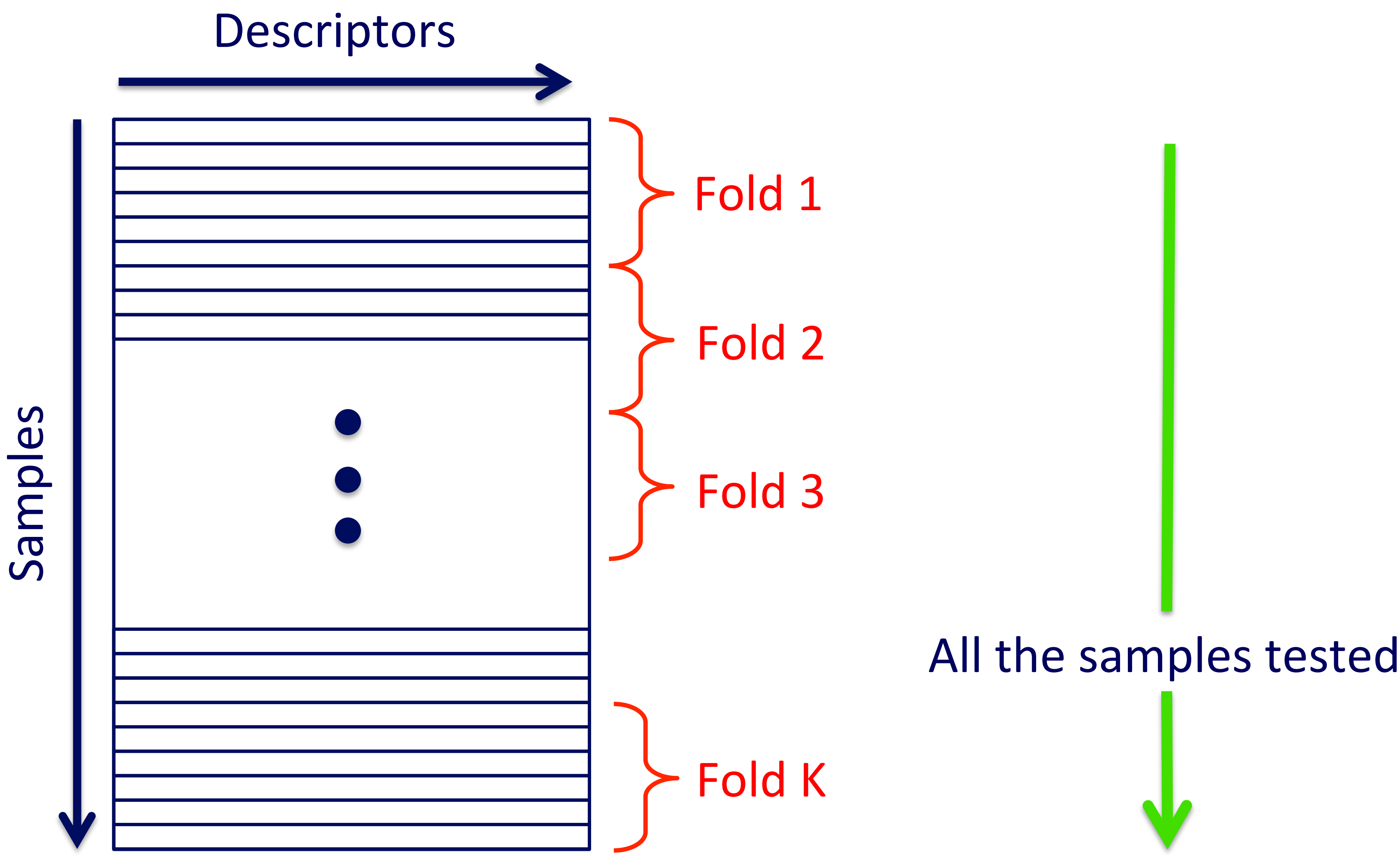


Validation

K fold cros-validation



K fold Cros-Validation



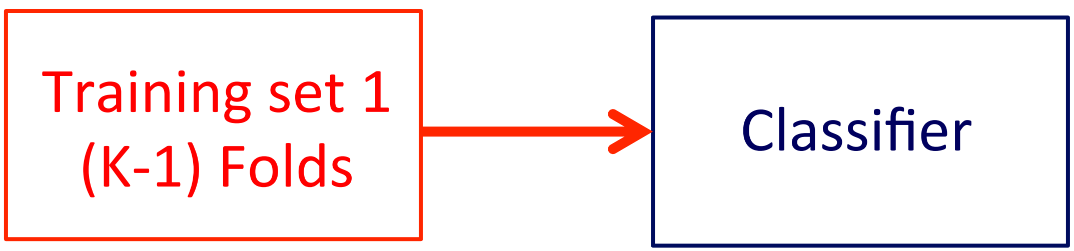
Parameters

K fold Cros-Validation

Training set 1
(K-1) Folds →

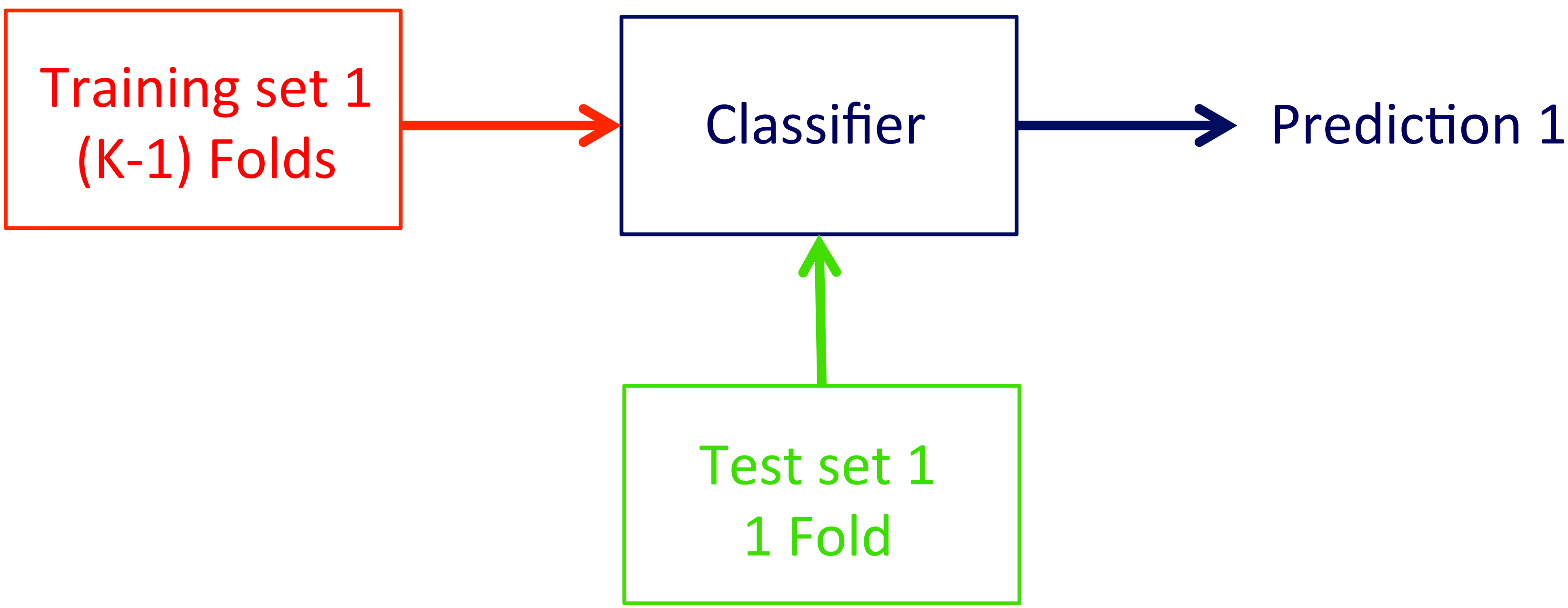
Parameters

K fold Cros-Validation



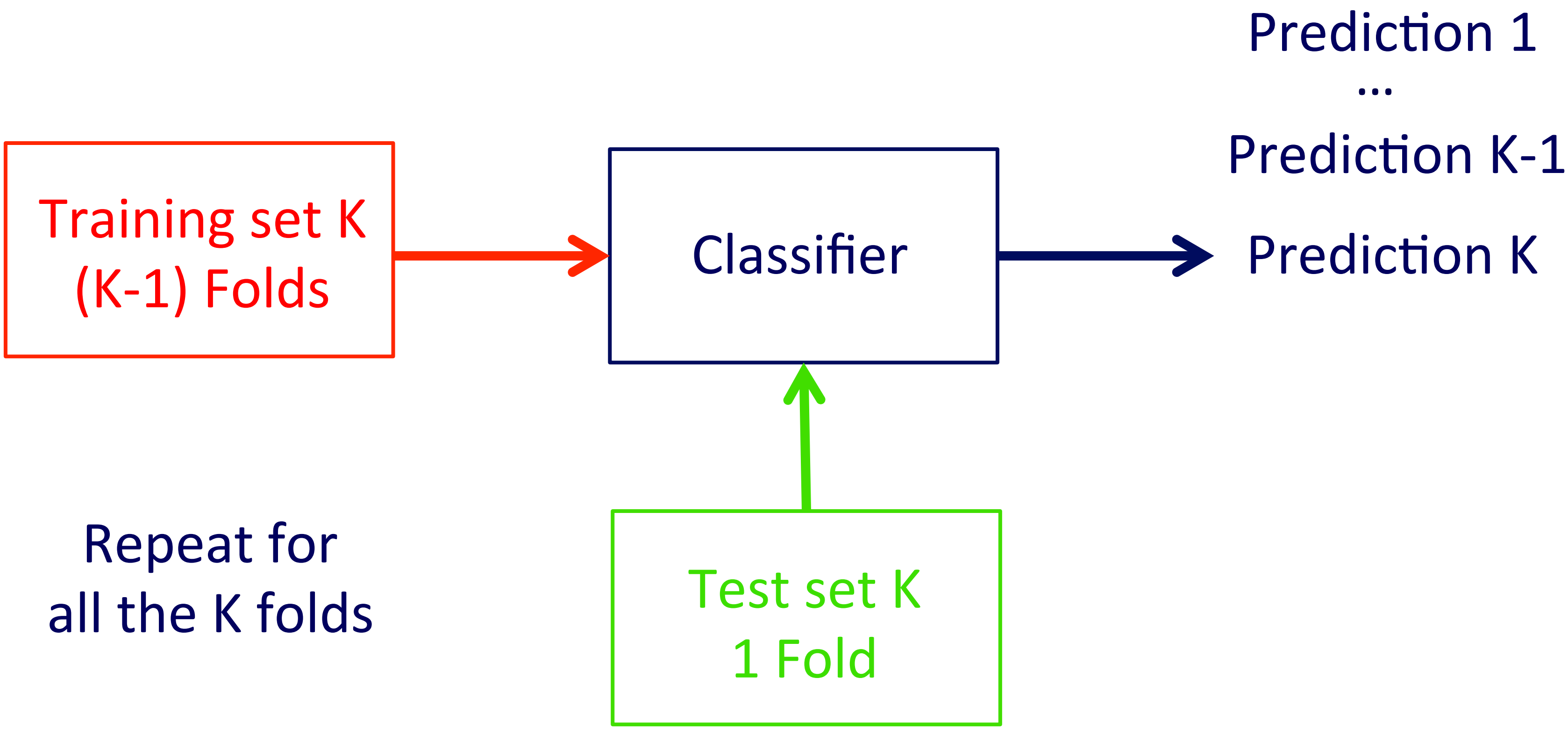
Parameters

K fold Cros-Validation

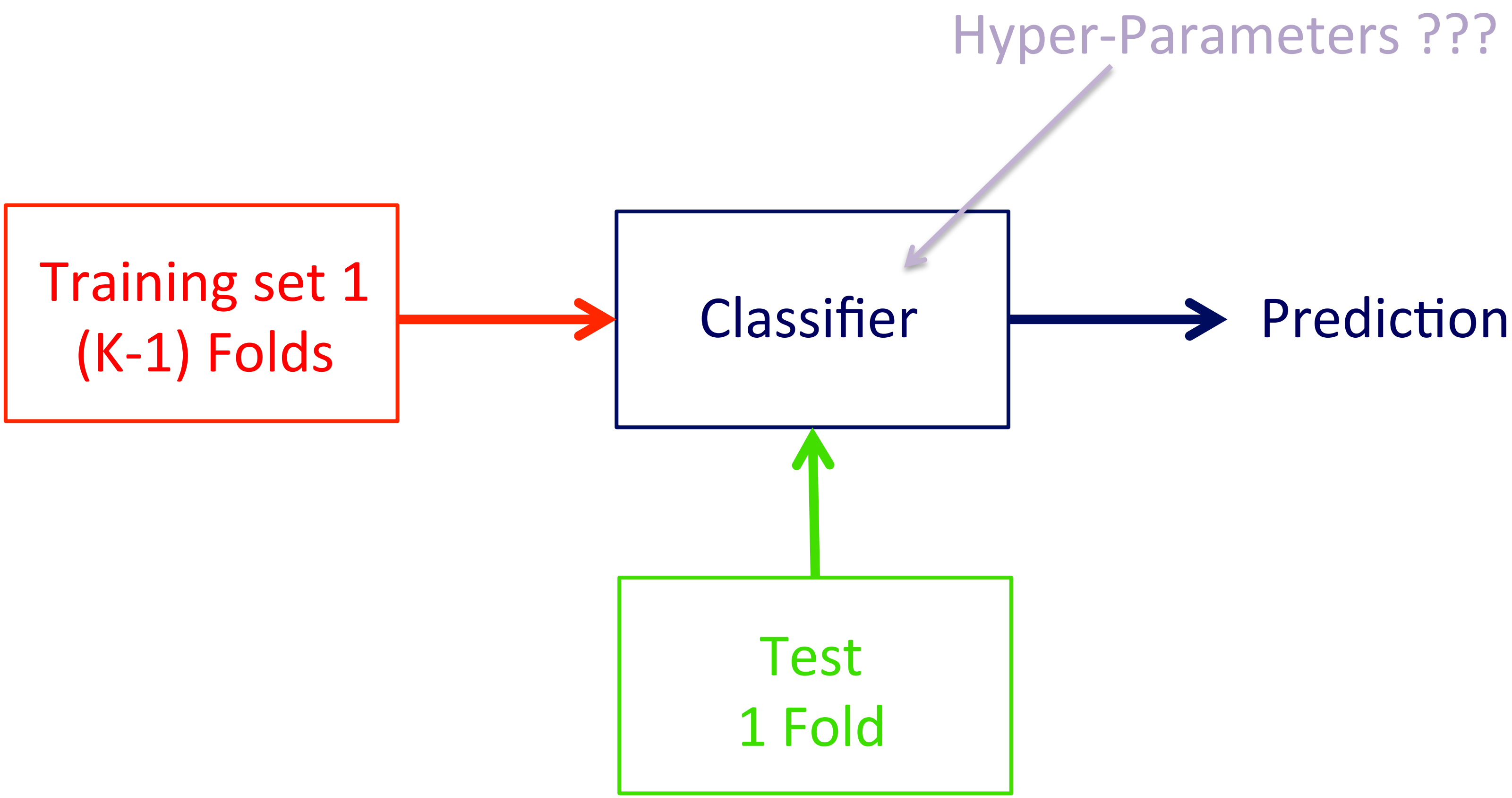


Parameters

K fold cros-validation

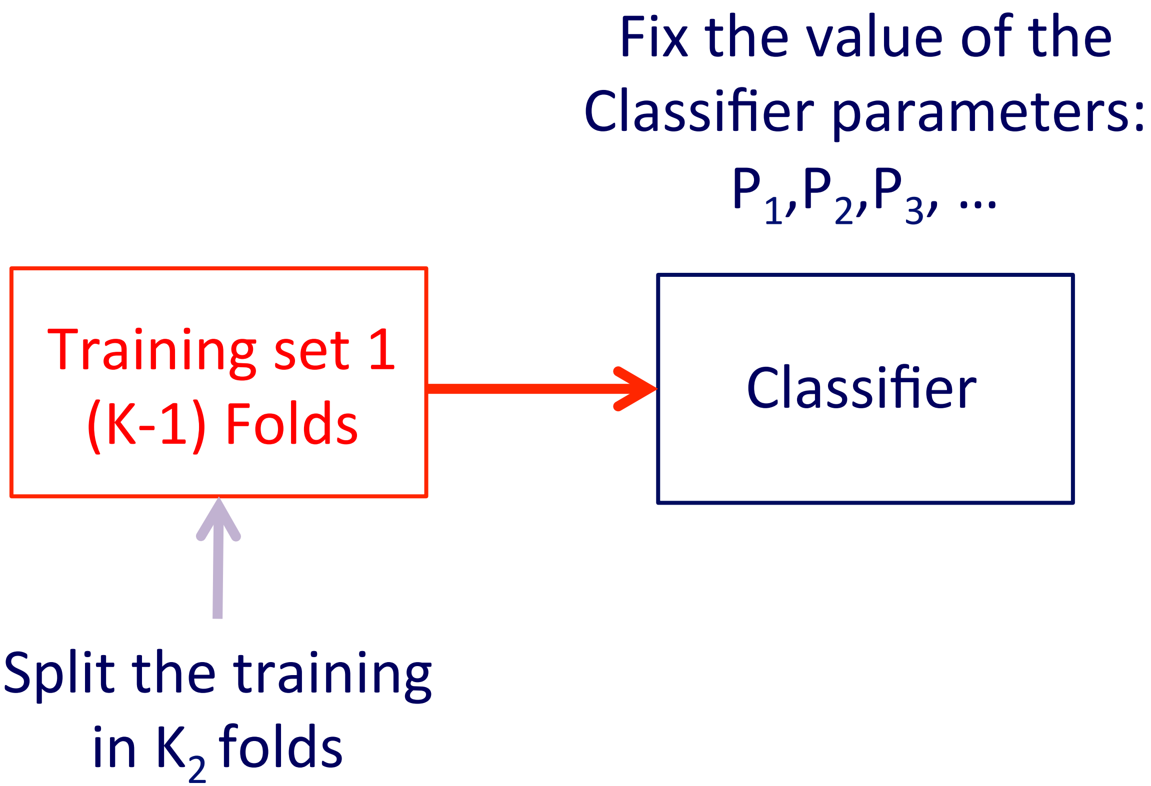


PARAMETERS TUNING



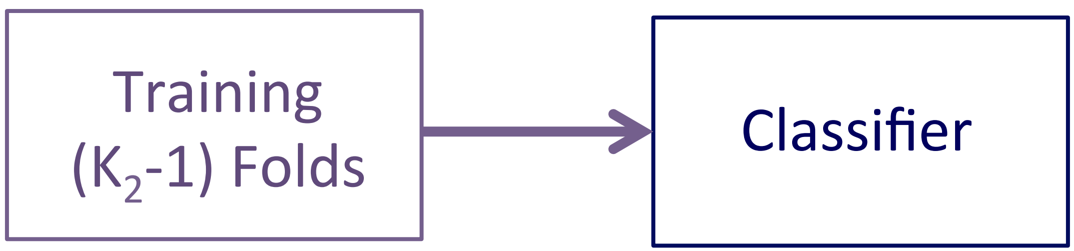
Parameters

K fold Cros-Validation on the training set



K FOLD CROS-VALIDATION ON THE TRAINING SET

Fix the value of the Classifier hyper-parameters: P_1, P_2, P_3, …

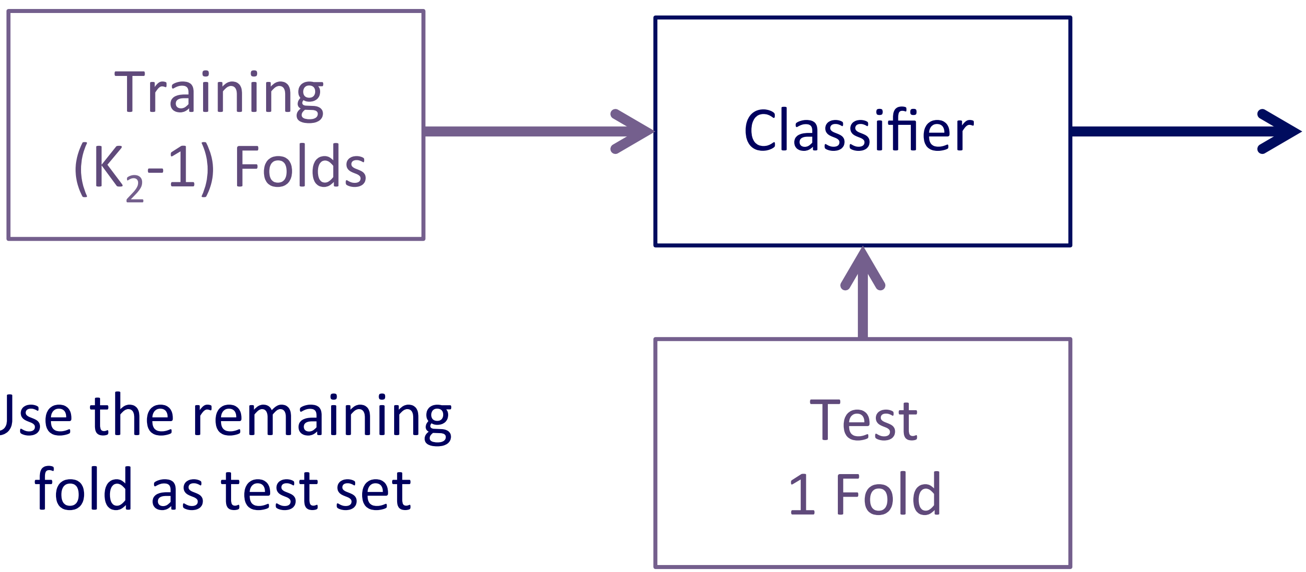


Use K_2 folds as Training set

Parameters

K fold Cros-Validation
on the training set

Fix the value of the
Classifier hyper-parameters:
$P_1, P_2, P_3, \ldots$



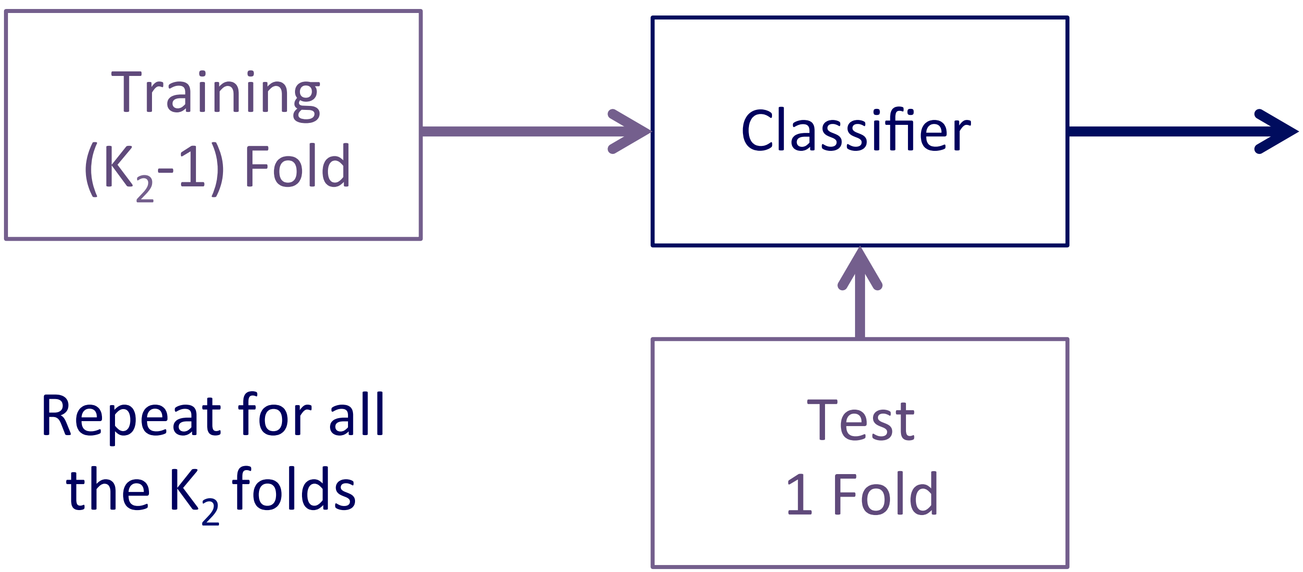
Prediction
on fold 1

Use the remaining
fold as test set

Parameters

K fold Cros-Validation
on the training set

Fix the value of the
Classifier hyper-parameters:
$P_1, P_2, P_3, \ldots$



Repeat for all
the K_2 folds

Prediction
on fold 1

Prediction
on fold 2

⋮

Prediction
on fold K_2

UCL Université catholique de Louvain

PARAMETERS

K FOLD CROS-VALIDATION ON THE TRAINING SET

Fix the value of the Classifier hyper-parameters: P_1, P_2, P_3, …

Training (K_2-1) Fold

Classifier

Test 1 Fold

Repeat for all the K_2 folds

Prediction on fold 1

Prediction on fold 2

⋮

Prediction on fold K_2

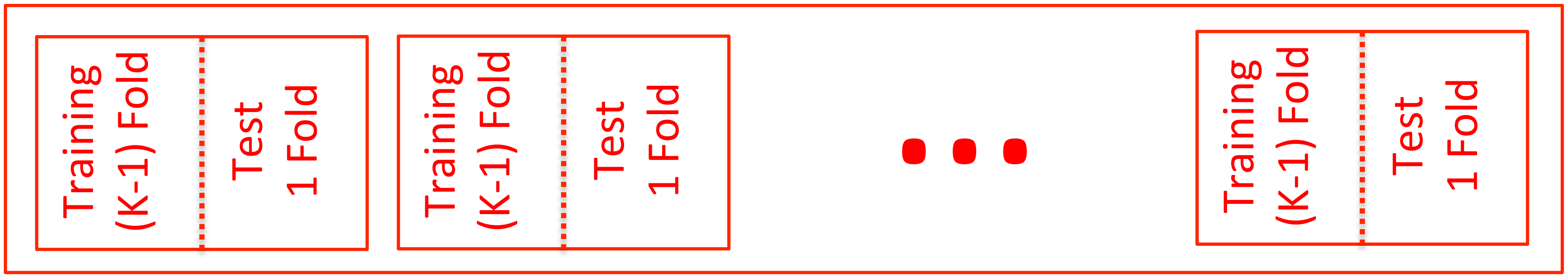
Average performance of the system using hyper-parameters: P_1, P_2, P_3, …

Parameters

K fold Cros-Validation on the training set

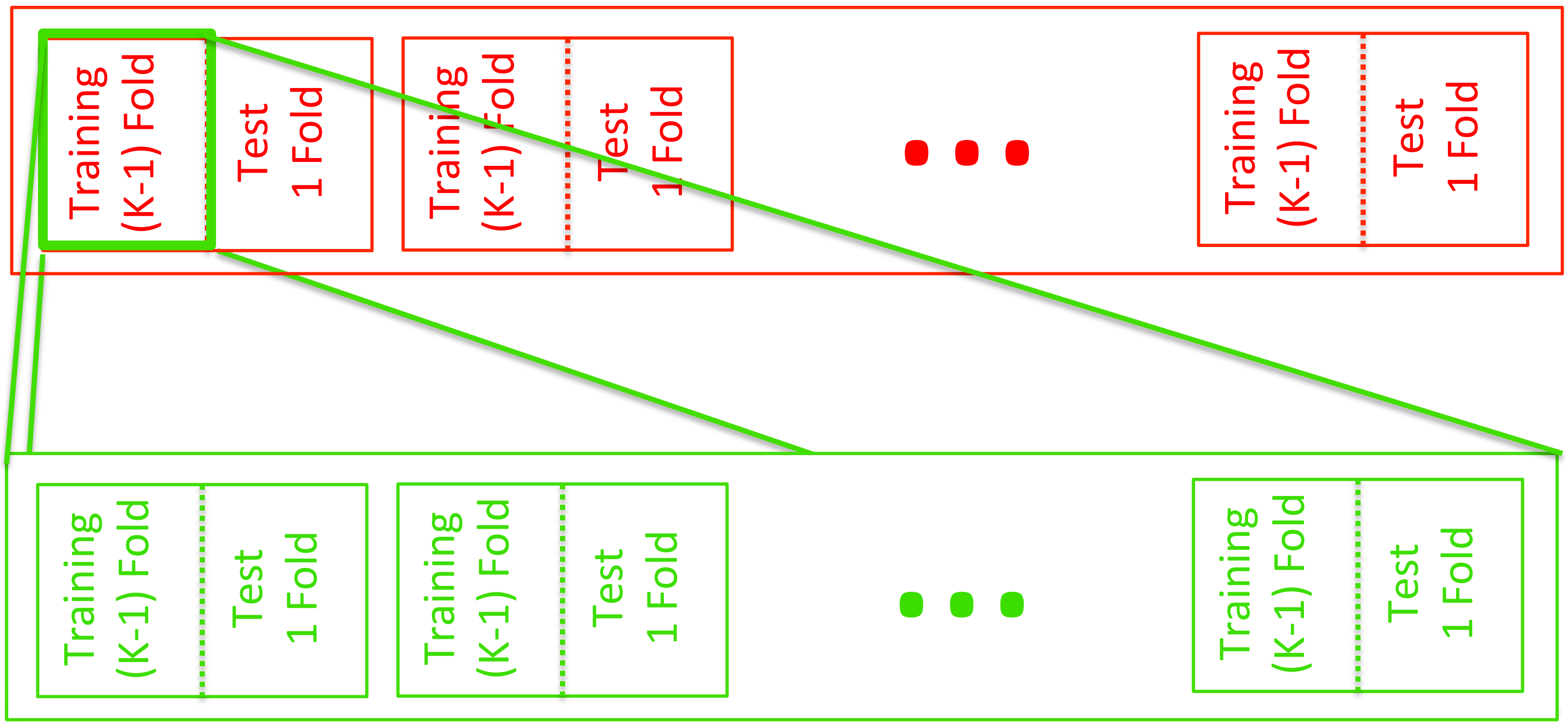
- Repeat the test with several hyper-parameters values
- Chose the set of hyper-parameters that provides the best performance
- Train the classifier using the chosen hyper-parameters

K fold cross validation

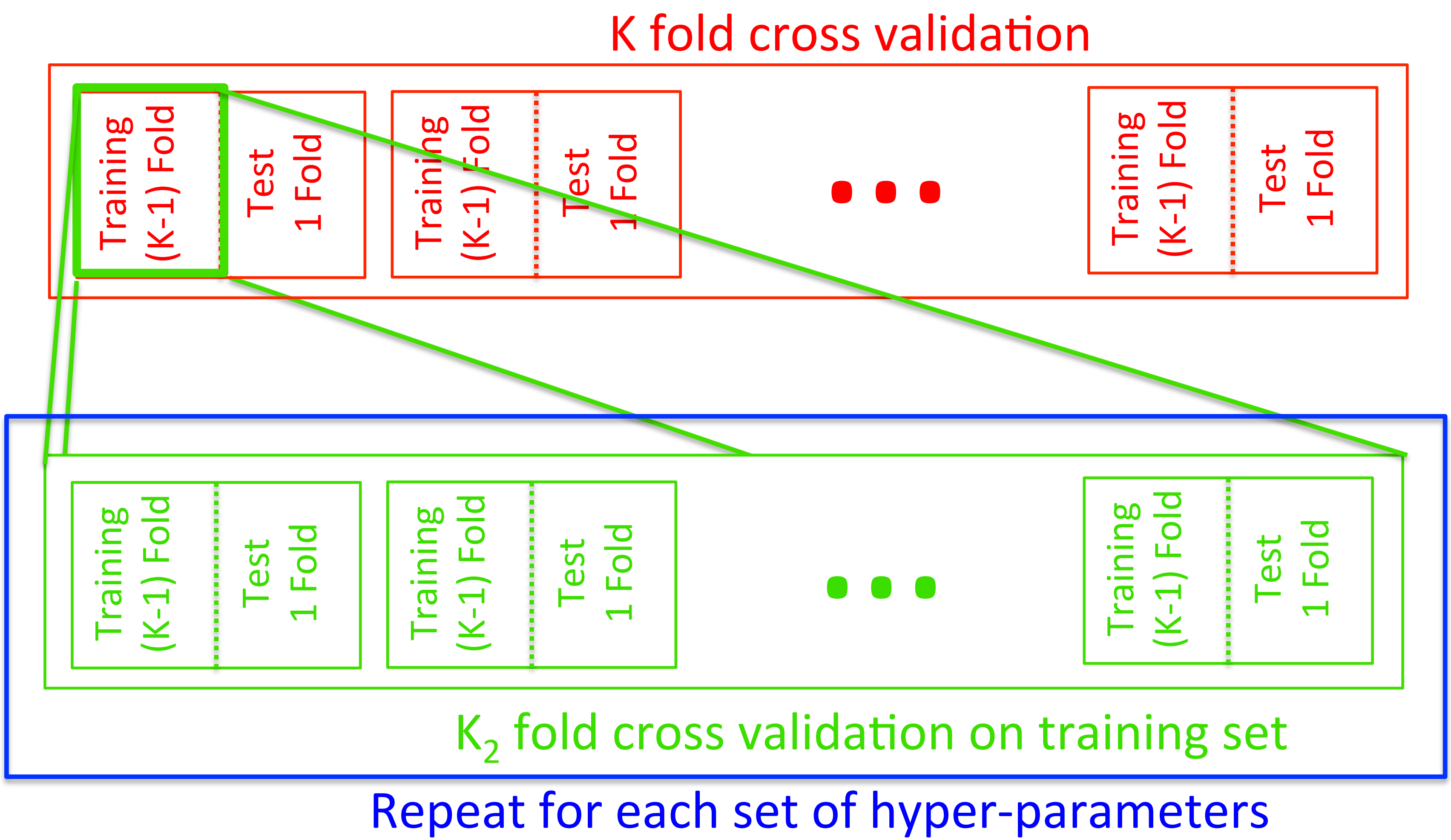


How to perfrom experiments

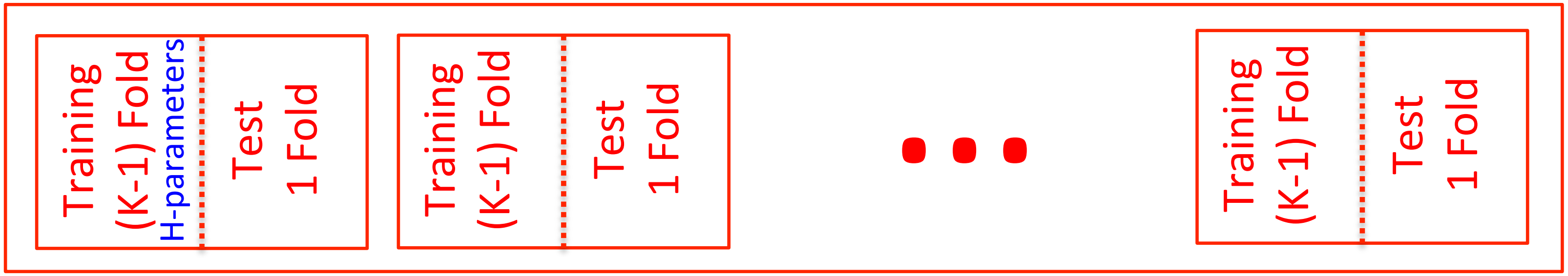
K fold cross validation



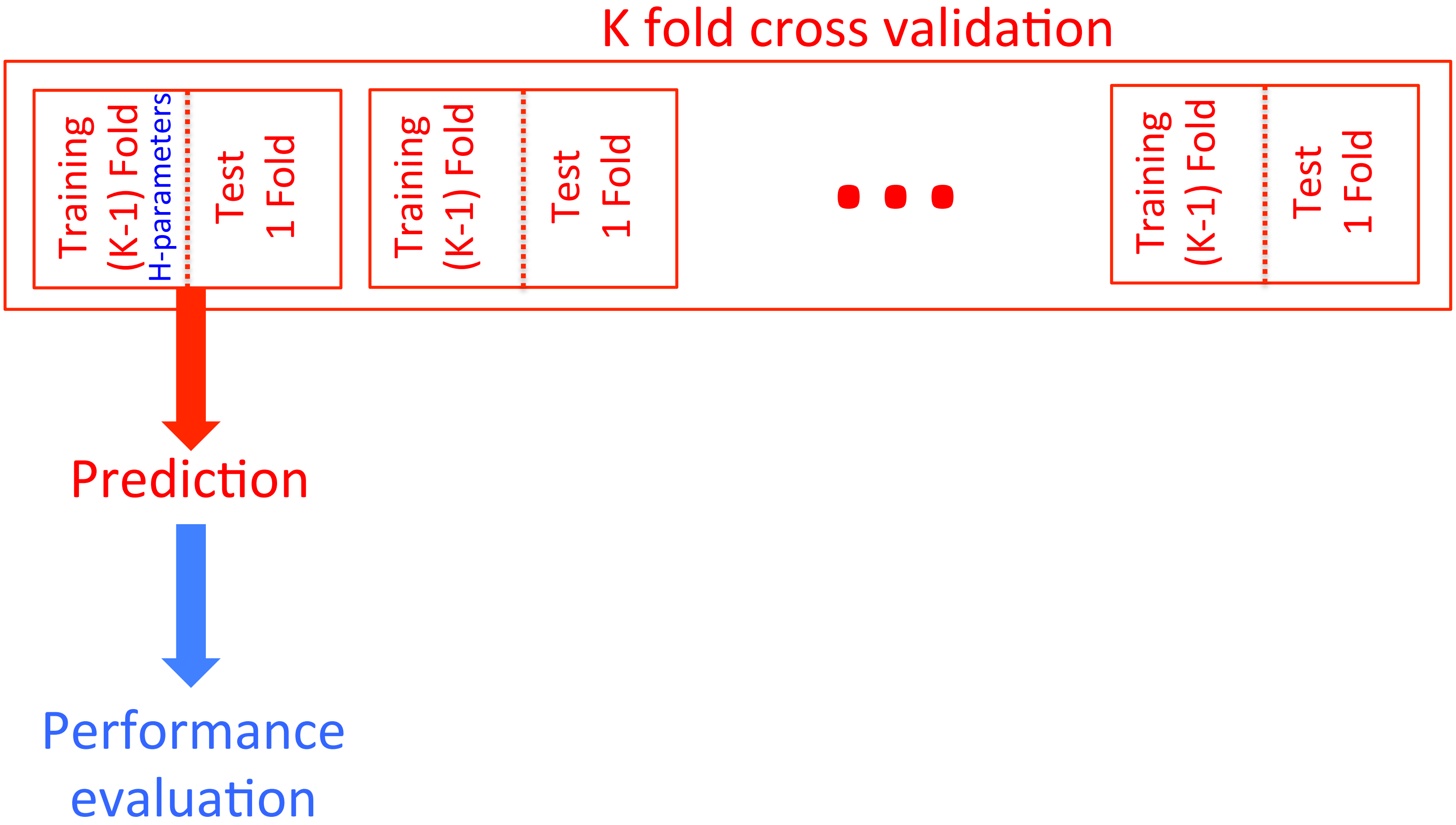
K_2 fold cross validation on training set


K fold cross validation
Training (K-1) Fold
Test 1 Fold
Training (K-1) Fold
Test 1 Fold
Training (K-1) Fold
Test 1 Fold
Training (K-1) Fold
Test 1 Fold
Training (K-1) Fold
Test 1 Fold
Training (K-1) Fold
Test 1 Fold
K2 fold cross validation on training set
Repeat for each set of hyper-parameters

K fold cross validation

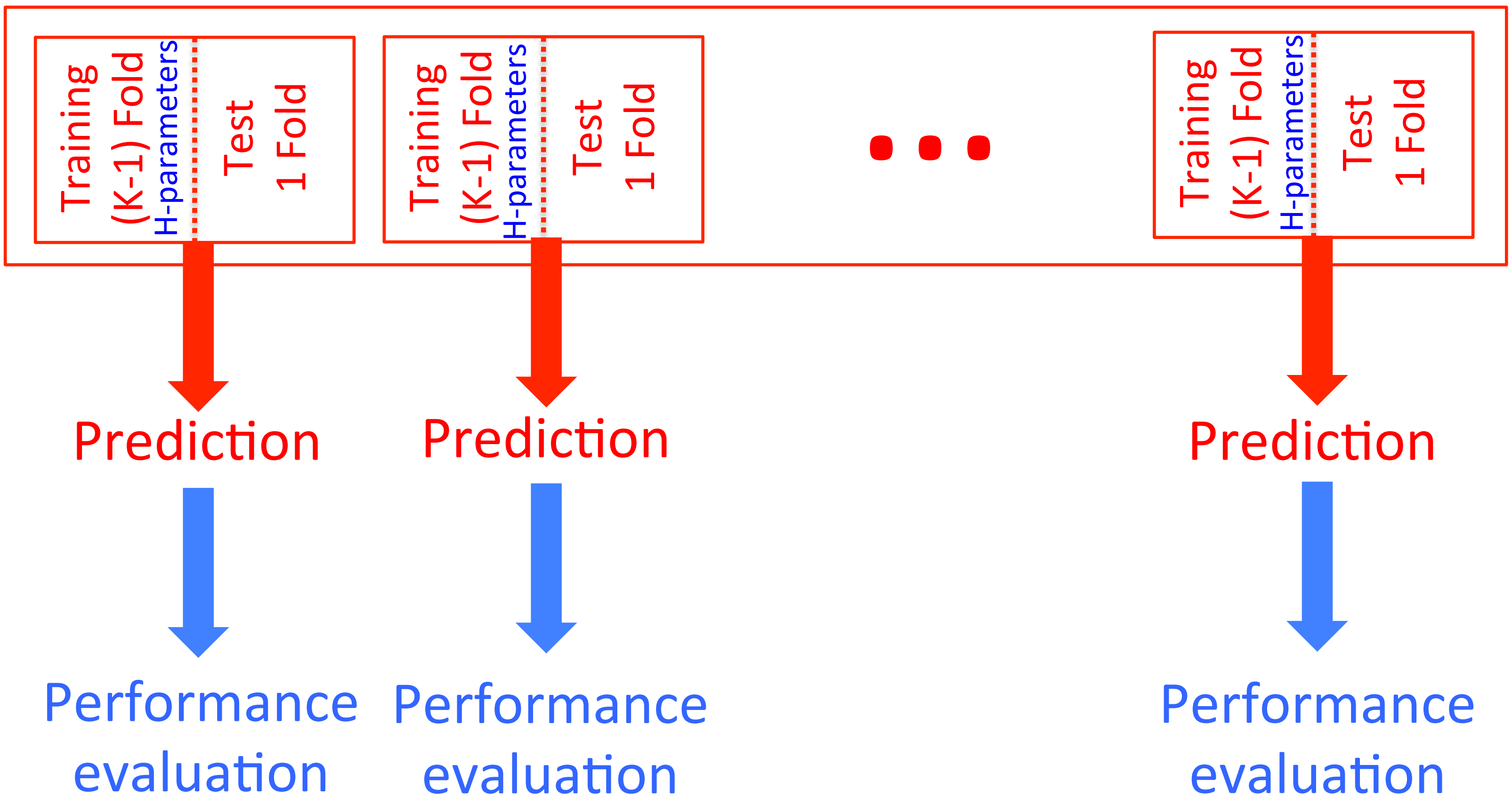


Experiments

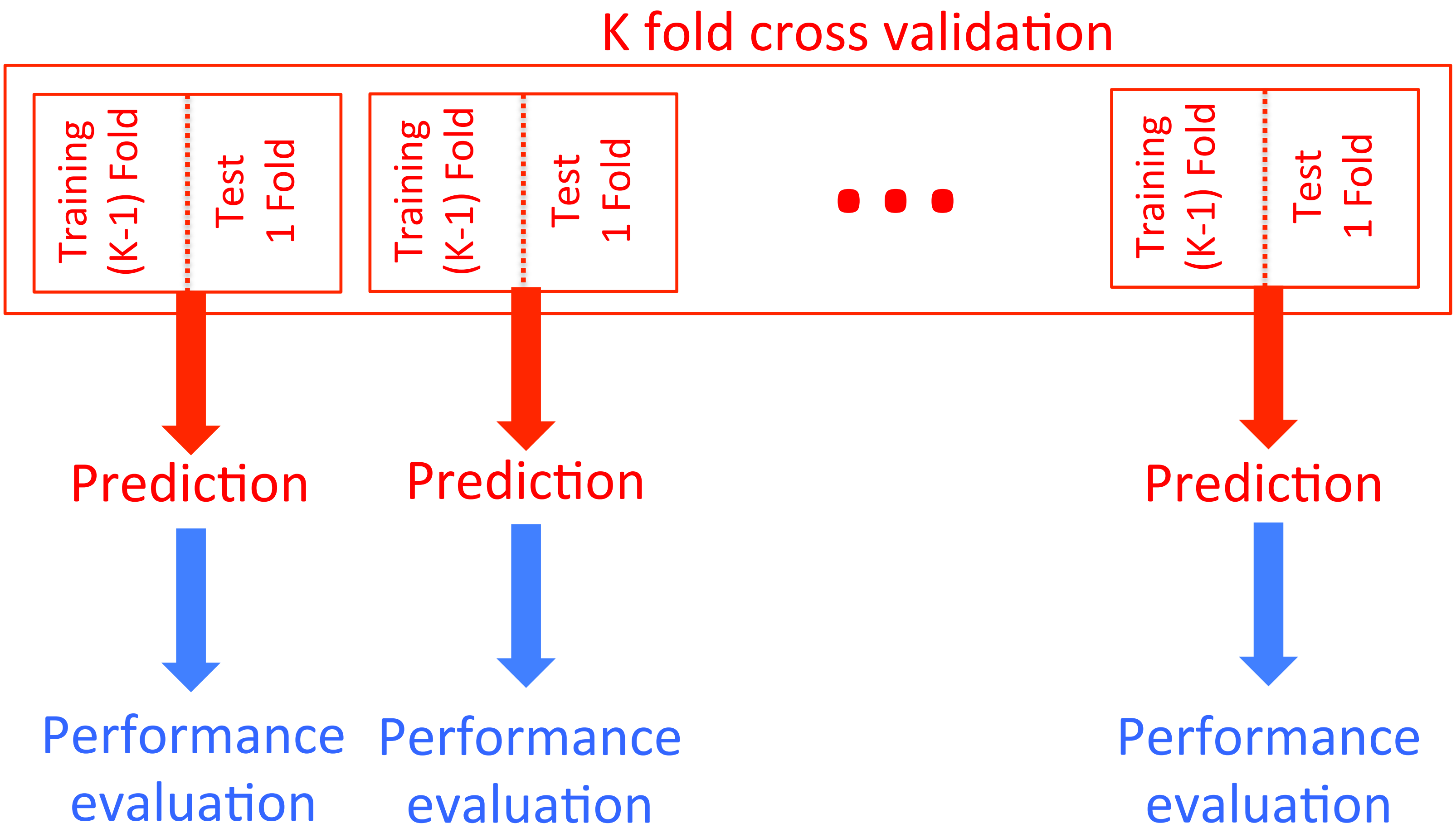


How to perfrom experiments

K fold cross validation

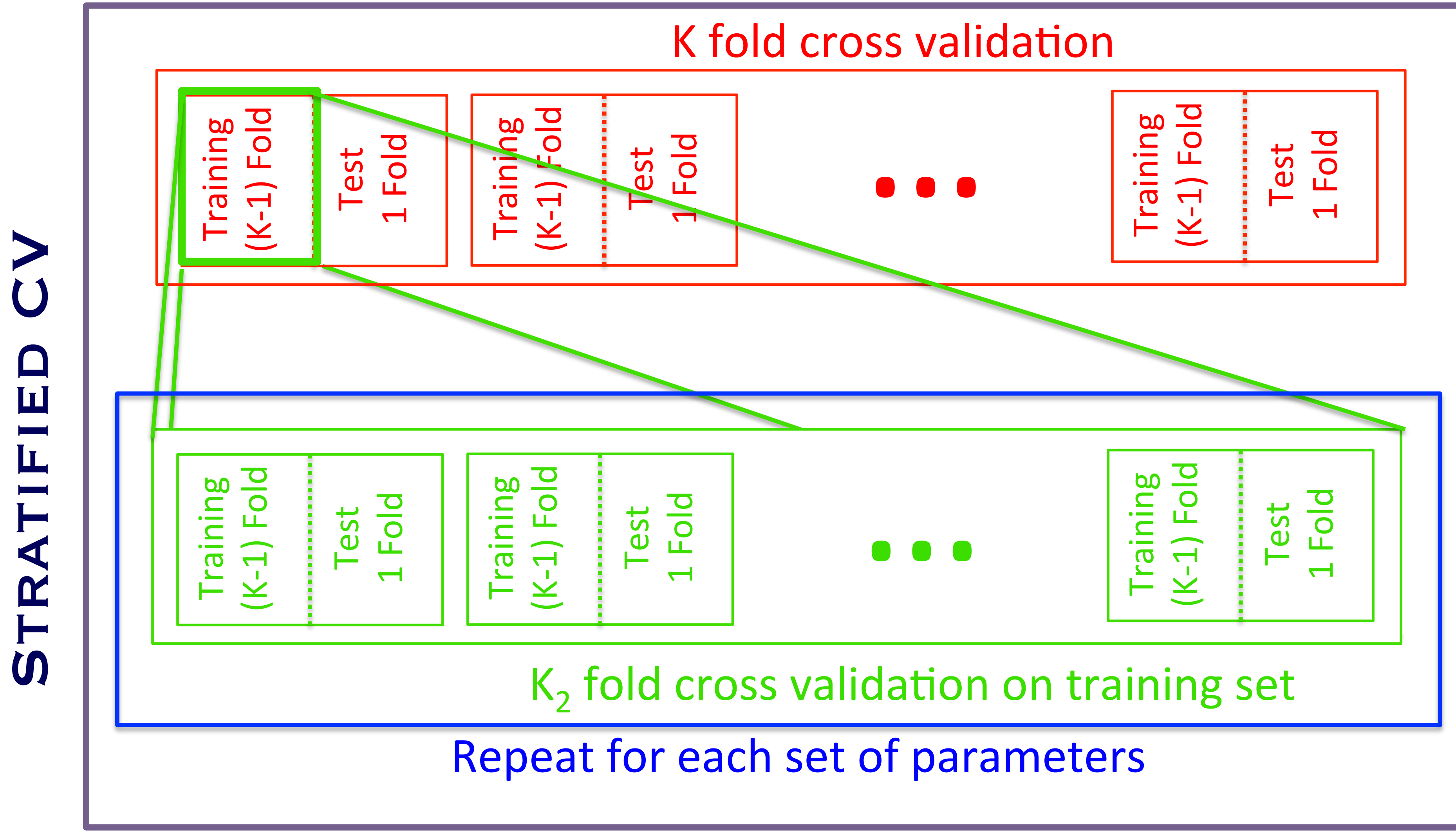


How to perfrom experiments



System performance are computed as average of performance among the folds

Experiments



Repeat for each layer of the stratification

Confusion Matrix

Ground Truth

Predicted Class

	p	n
$\tilde{P}$	**T**rue **P**ositive	**F**alse **P**ositive
$\tilde{N}$	**F**alse **N**egative	**T**rue **N**egative
	P	N

Accuracy

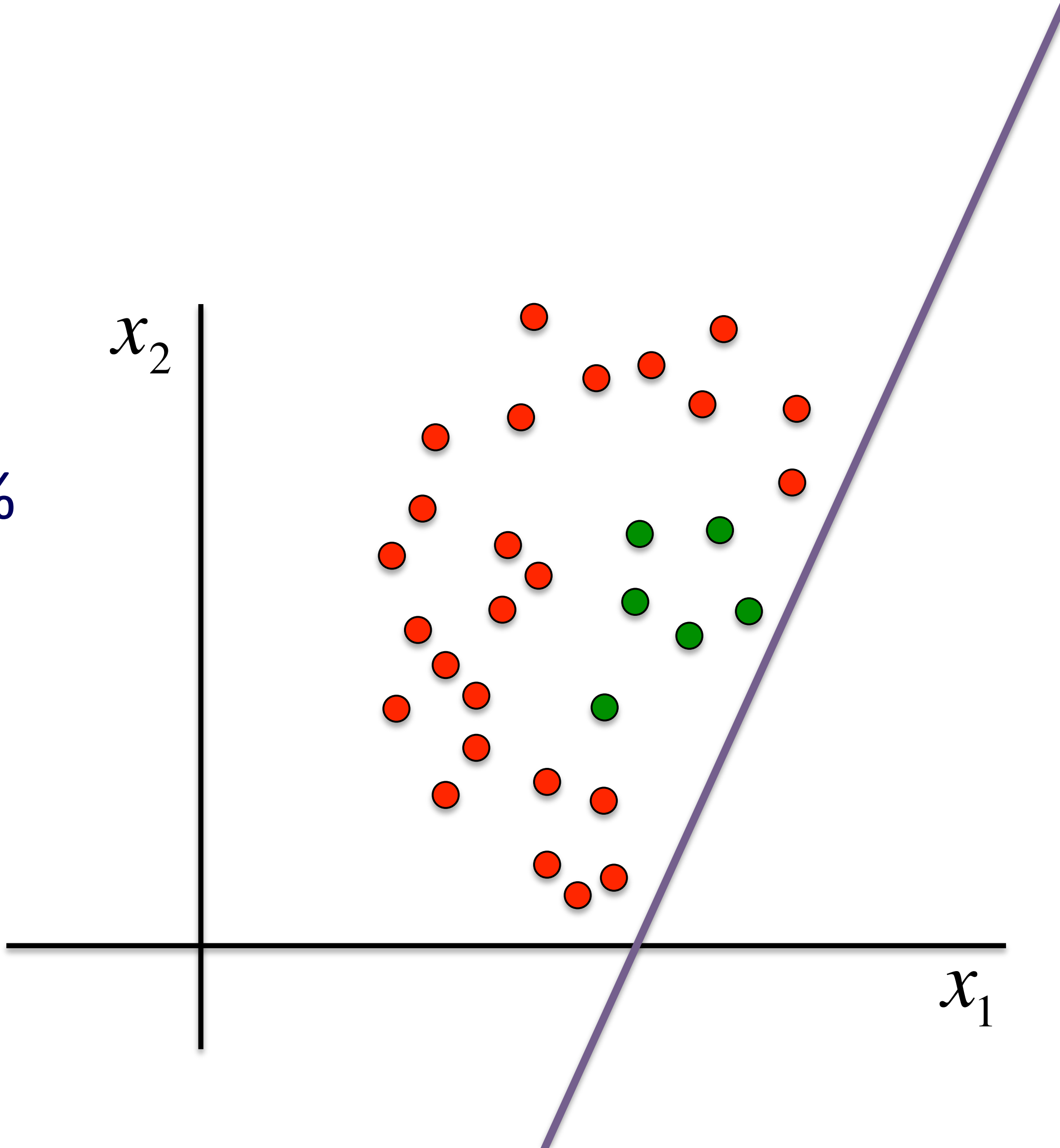
$$Acc = \frac{TP+TN}{P+N}$$

	p	n
$\tilde{P}$	**T**rue **P**ositive	**F**alse **P**ositive
$\tilde{N}$	**F**alse **N**egative	**T**rue **N**egative
	P	N

The number of samples correctly classified over the total number of samples

Accuracy Limitations

Accuracy = 80%



Other Metrics

$$\text{Recall} = \frac{TP}{P}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

	p	n
$\tilde{P}$	**T**rue **P**ositive	**F**alse **P**ositive
$\tilde{N}$	**F**alse **N**egative	**T**rue **N**egative
	P	N

Other Metrics

$$\text{G-mean} = \left(\frac{TP}{P} \cdot \frac{TN}{N} \right)^{\frac{1}{2}}$$

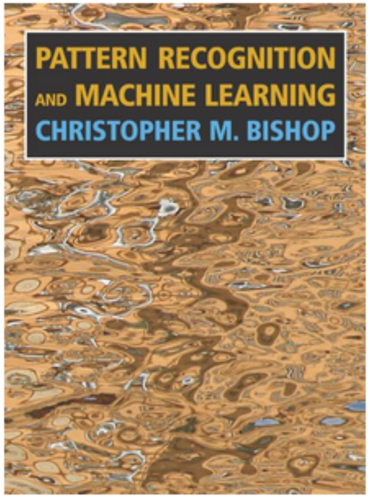
$$\text{Class acc} = \frac{\frac{TP}{P} + \frac{TN}{N}}{2}$$

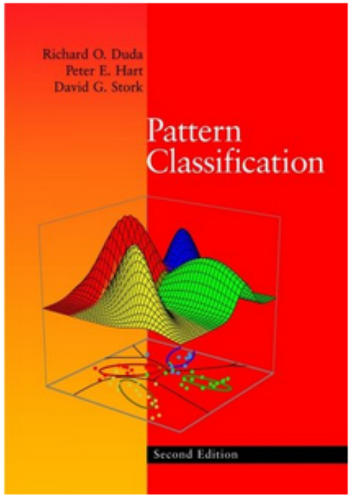
$$\text{F-measure} = 2 \frac{\text{Precision} x \text{Recall}}{\text{Precision} + \text{Recall}}$$

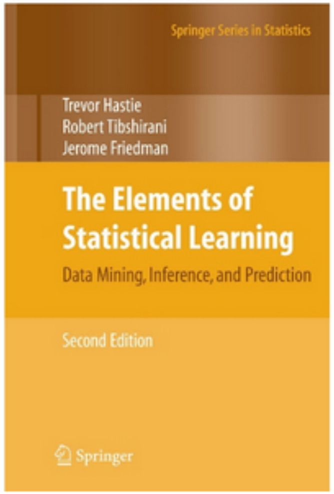
	p	n
$\tilde{P}$	**T**rue **P**ositive	**F**alse **P**ositive
$\tilde{N}$	**F**alse **N**egative	**T**rue **N**egative
	P	N

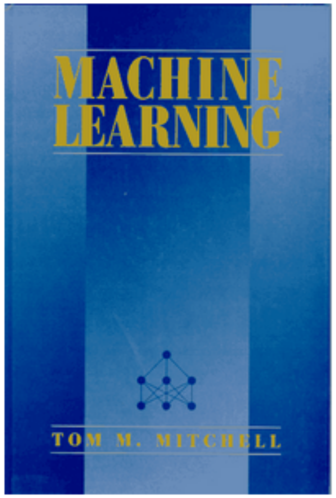
Books

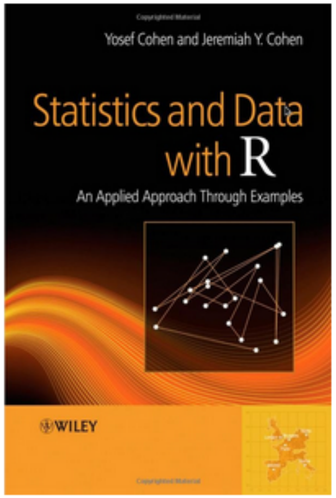
- Pattern Recognition and Machine Learning; C.M.Bishop
- Pattern Classification; R.O. Duda, P.E. Hart, D.G. Stork
- Machine Learning; T.M.Mitchell
- Statistics and Data with R; Y. Cohen, J.Y. Cohen











Thank you for your attention