POLITECNICO DI MILANO

Support Vector Machines: a gentle introduction

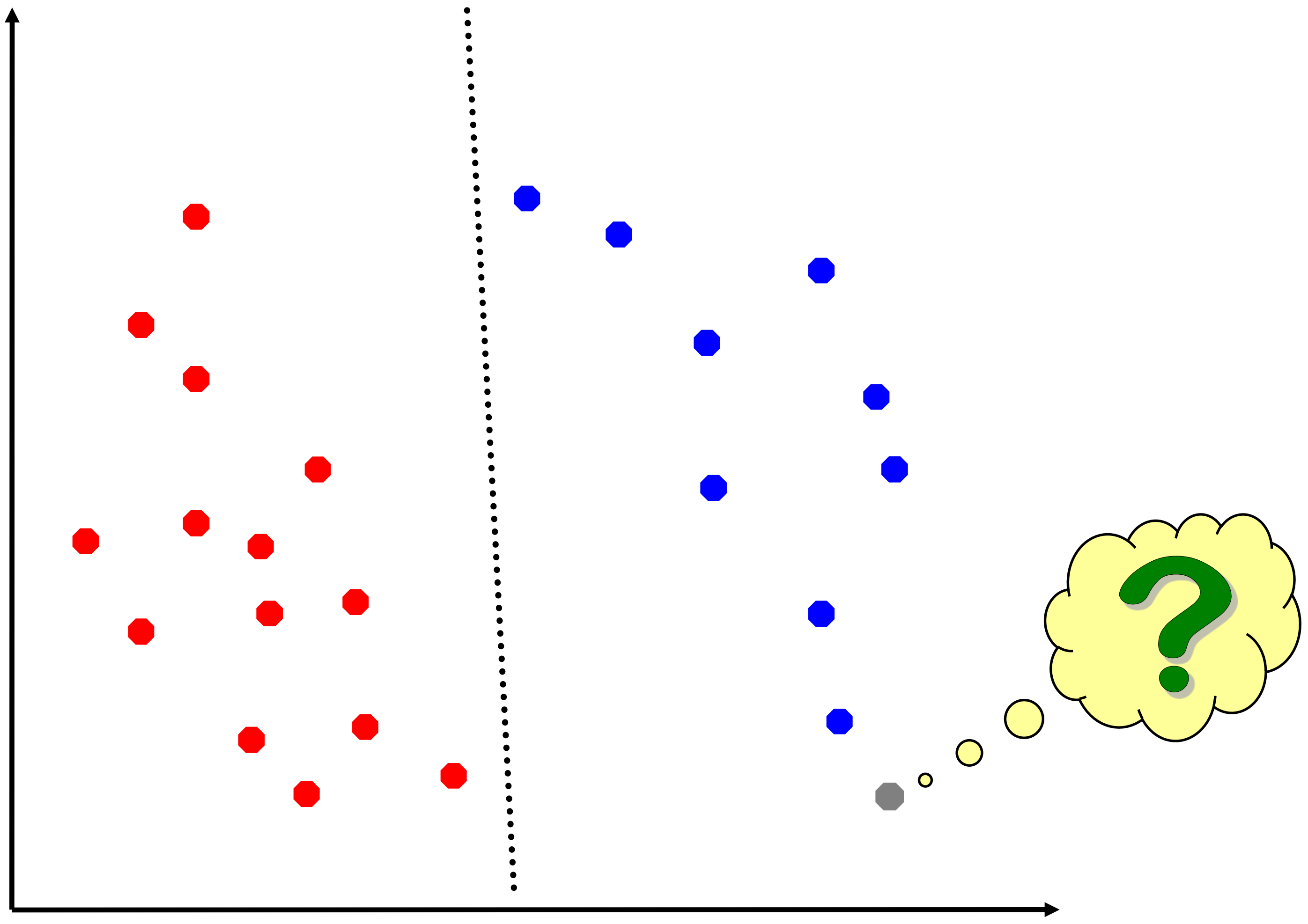
Data Mining and Text Mining (UIC 583 @ Politecnico di Milano)

Daniele Loiacono

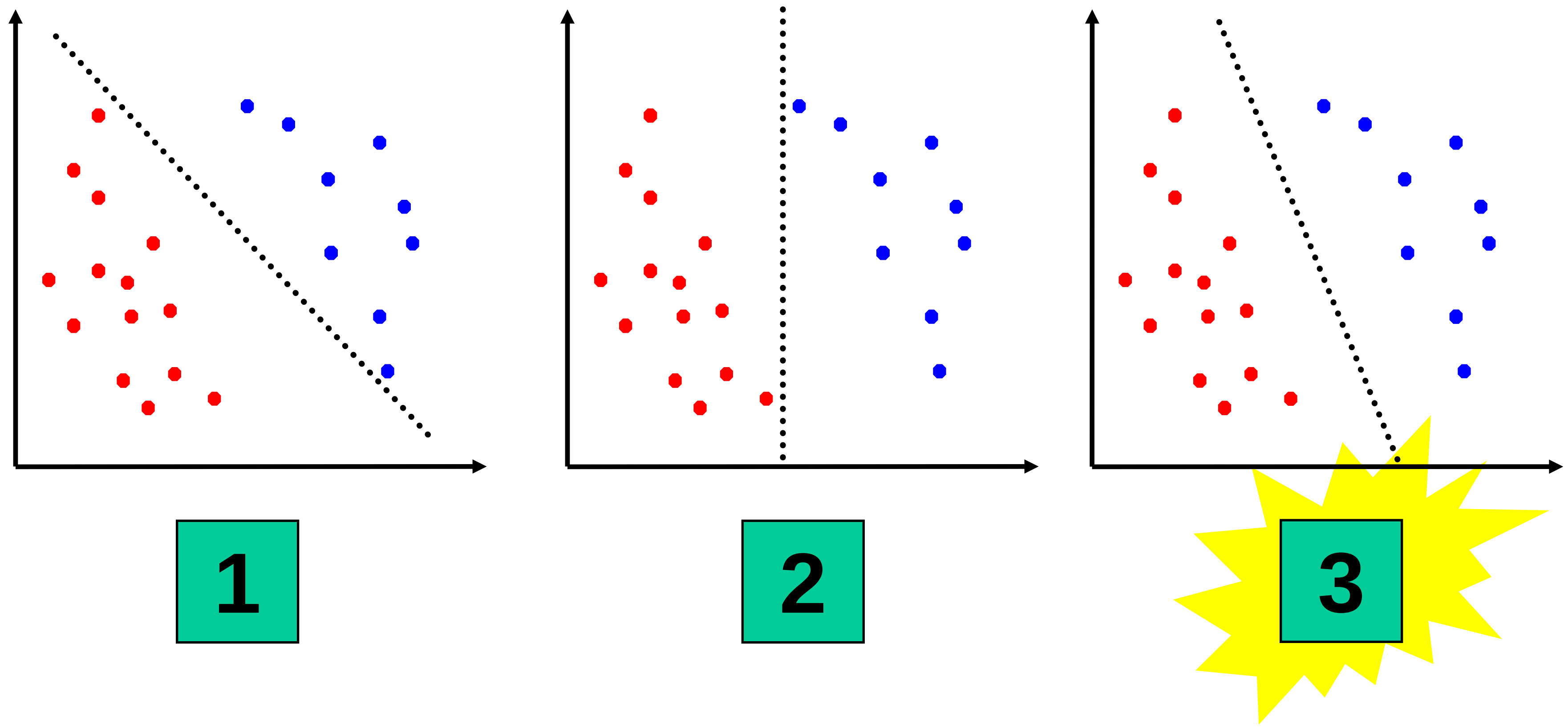
Part 1: What are SVMs?

- The problem
- Separating Hyperplane
- The optimization problem
- Derivation of the dual problem and Support Vector Expansion
- The Soft-Margin optimization problem

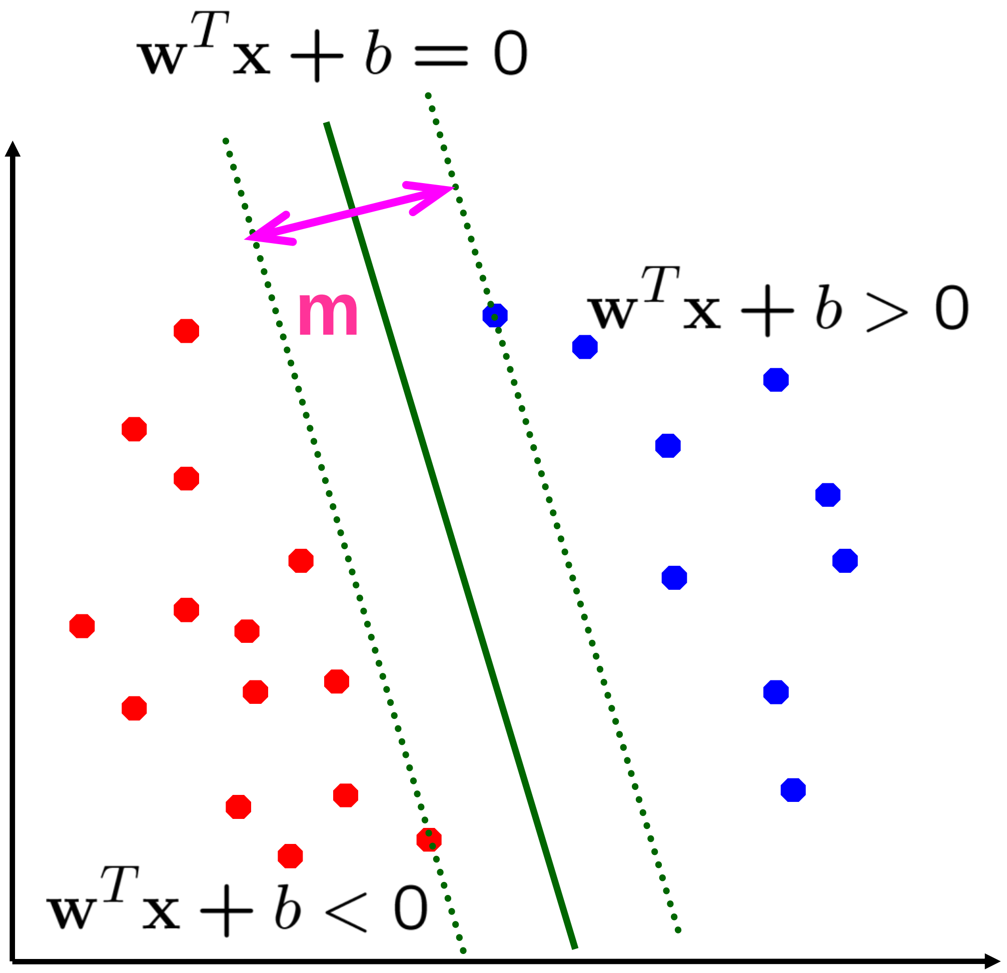
What is all about?



Guess the best!



Separating Hyperplane



$\mathbf{w}^T\mathbf{x} + b = 0$
m
$\mathbf{w}^T\mathbf{x} + b > 0$
$\mathbf{w}^T\mathbf{x} + b < 0$

Separating Hyperplane

- Scale problem

$$\forall c > 0, \{\mathbf{w}^T\mathbf{x} + b = 0\} \iff \{c\mathbf{w}^T\mathbf{x} + cb = 0\}$$

- **Canonical** hyperplane

$$\min_{x_i} |w^T x_i + b| = 1$$

- **Margin** of the hyperplane

$$m = \frac{2}{||\mathbf{w}||}$$

The optimization problem

- More formally the problem is:

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \forall i$$

- Is an optimization problem with inequality constraints…

Derivation of the dual problem

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

- L has to minimized w.r.t. the **primal variables w** and b and maximized with respect to the **dual variables** $\alpha_i >= 0$

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial}{\partial b} \mathcal{L} = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

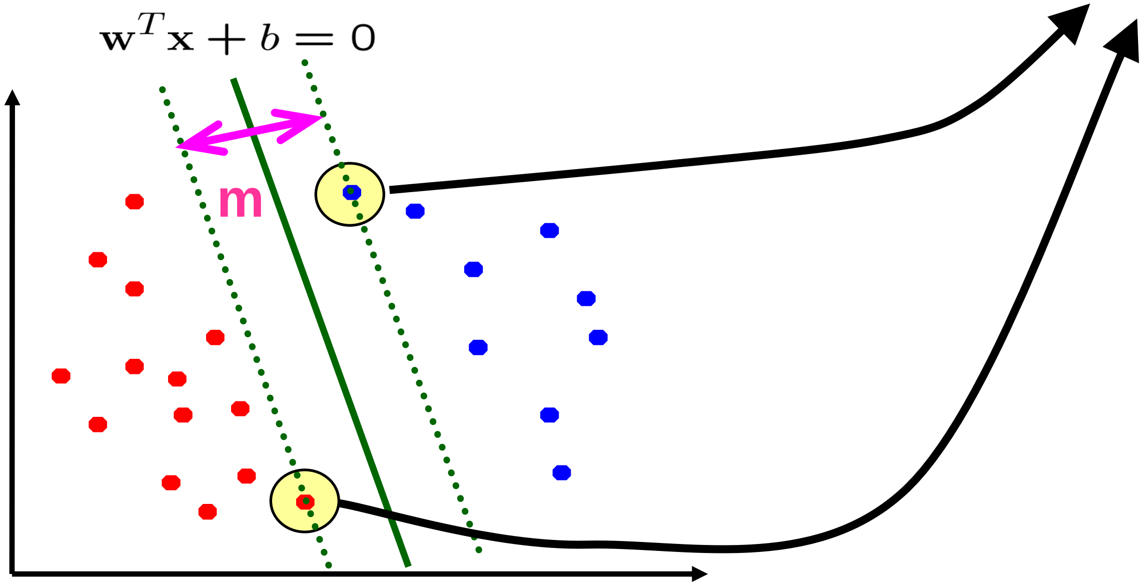
- Substitute both in L to get the **dual problem**

Support Vectors

$$\boxed{\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i}$$

where for all $i = 1, \cdots, m$

$$\begin{cases} y_i(\mathbf{w}^T\mathbf{x}_i + b) > 1 & \alpha_i = 0 \Rightarrow x_i \text{ is irrilevant} \\ y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1 & \alpha_i \neq 0 \Rightarrow x_i \text{ is a } \underline{\textit{Support Vector}} \end{cases}$$



The dual problem

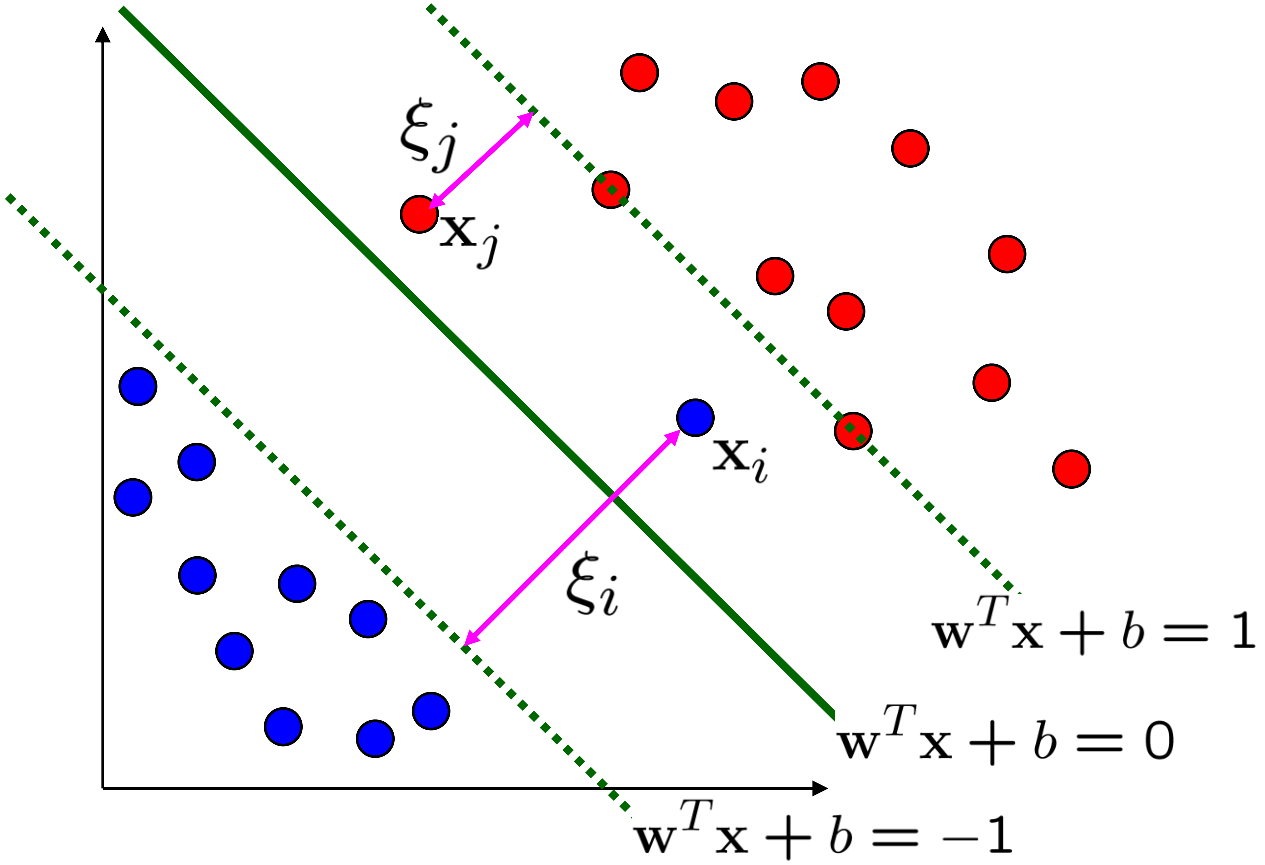
- The value of α_i can be found solving the following quadratic optimization problem (the dual optimization problem):

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Non-linearly Separable Problem

- In practice problem are not often linearly separable
- In this case we allow "error" ξ_i in classification:



ξ_j
$\mathbf{x}_j$
$\mathbf{x}_i$
ξ_i
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

Soft-Margin optimization problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i > 0 \quad \forall i$$

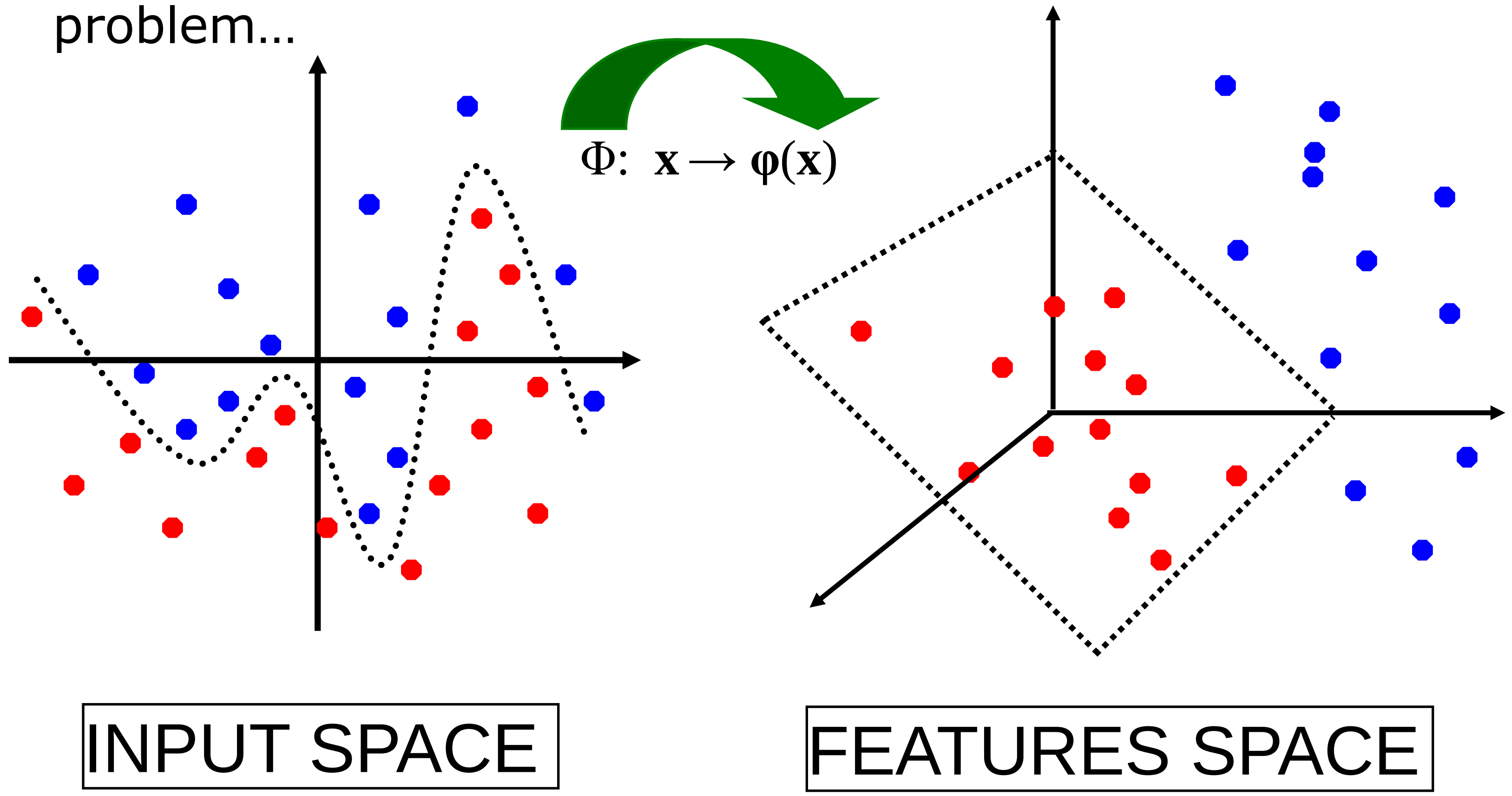
- ξ_i are "slack variables" in optimization
- *C* is a tradeoff parameter between error and margin

Part 2: SVMs in a non linear world

- Extension to non linear classification problem
- Mastering the Kernel Trick

Non-linearly separable problems

- So far we considered only (almost) linearly separable problems
- In practice we have to deal with non-linearly separable problem…



The Kernel Trick

- But a suitable features space have often **very high dimensionality** (too expensive for computation)
- We use the Kernel trick! Recall the dual problem:

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- We only need to know the dot product in the features space
- We define the **kernel function** as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

and

$$\mathbf{x}_i^T \mathbf{x}_j \rightarrow K(\mathbf{x}_i, \mathbf{x}_j)$$

An Example for kernel

- Suppose φ(.) is given as follows

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

- An inner product in the feature space is

$$\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) \rangle = (1 + x_1y_1 + x_2y_2)^2$$

- So, if we define the kernel function as follows, there is no need to carry out φ(.) explicitly

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- In general even some high dimensional features space admit an easy to compute kernel!

Part 3: Why does SVMs work?

A naïve introduction to statistical learning (VC) theory:

- ❑ Risk Minimization
- ❑ Uniform convergence
- ❑ VC-dimension
- ❑ Structural Risk Minimization
- ❑ The VC theory and SVMs

Risk Minimization

- We want learn a decision function f: X → {-1,+1} from a set of samples $\{(x_1,y_1),\dots,(x_n,y_n)\}$ generated i.i.d. from P(x, y)
- The expected misclassification error on a test set will be:

$$R[f] = \int \frac{1}{2} |f(x) - y| \, dP(x,y)$$

- Our goal is the minimization of R[f] **(Risk Minimization)**
- But P(x,y) is unknown, we can only minimize the error on the training set (**Empirical Risk**):

$$R_{emp}[f] = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} |f(x) - y)|$$

Does it converge?

- At least we want to prove that:

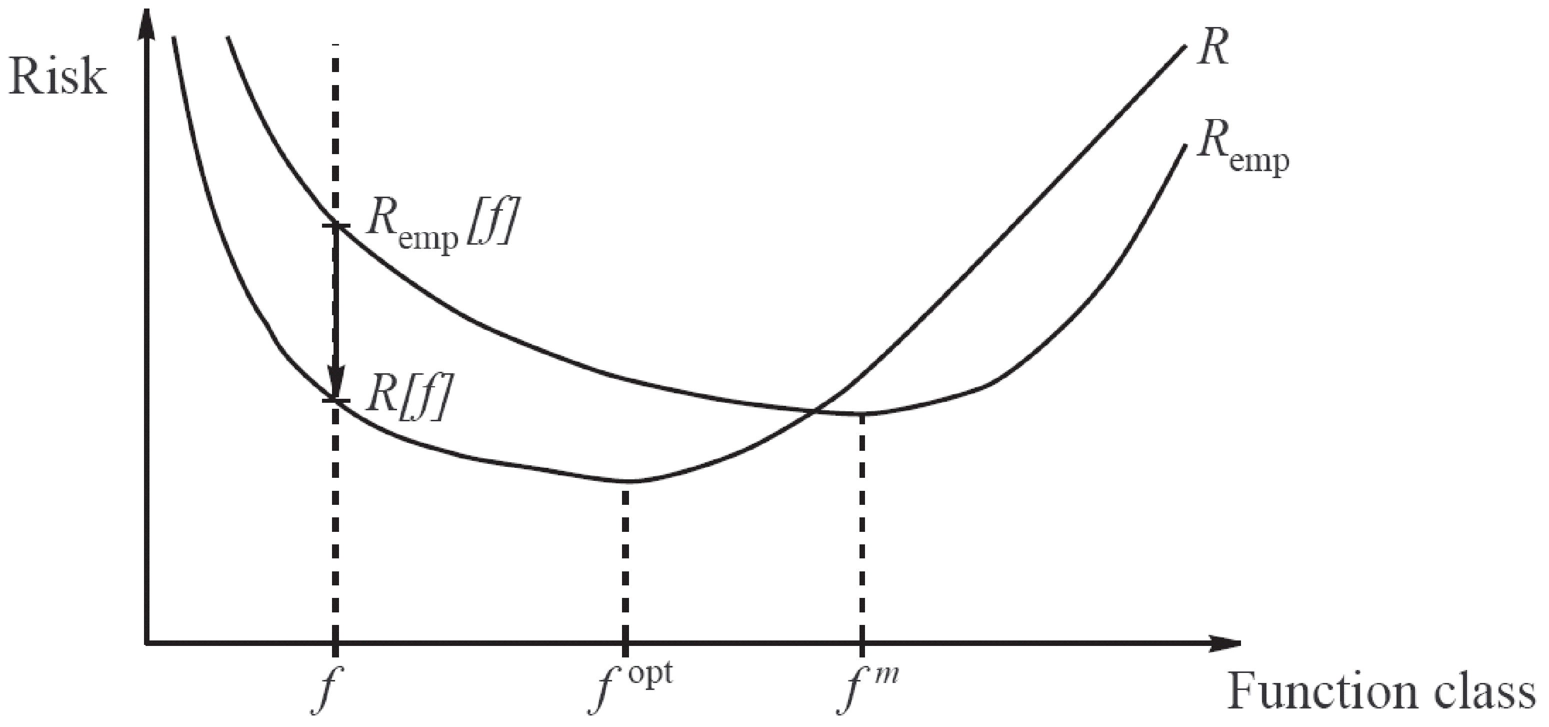
$$\lim_{n\to\infty} R_{emp}[f] = R[f]$$

- Sketch of the proof
  - loss: $\xi_i = \frac{1}{2}|f(x_i) - y_i|$
  - ξ_i as independent Bernoulli trials
  - for the law of large numbers the empirical mean (the Empirical Risk) will converge to the expected value of ξ_i (the Risk, R[f])
  - Chernoff's Bound:

$$P\left(\left|\frac{1}{n}\sum_{i=1}^{n}\xi_i - E[\xi]\right| \geq \varepsilon\right) \leq 2e^{-2n\varepsilon^2}$$

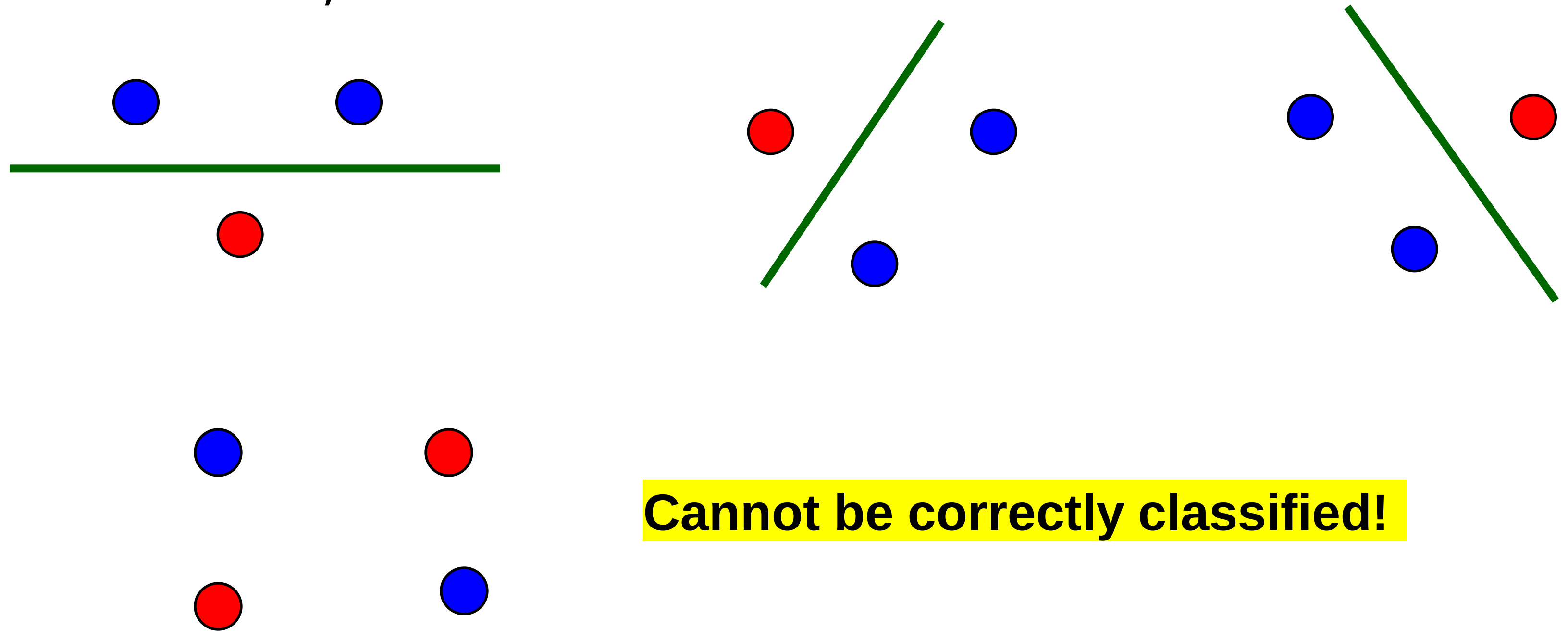
The sad truth!

- ξ_i are not independent if the function class is not fixed
- In a more general case we would need an uniform convergence
- How can we find a bound? Statistical Learning (VC) Theory



VC-dimension

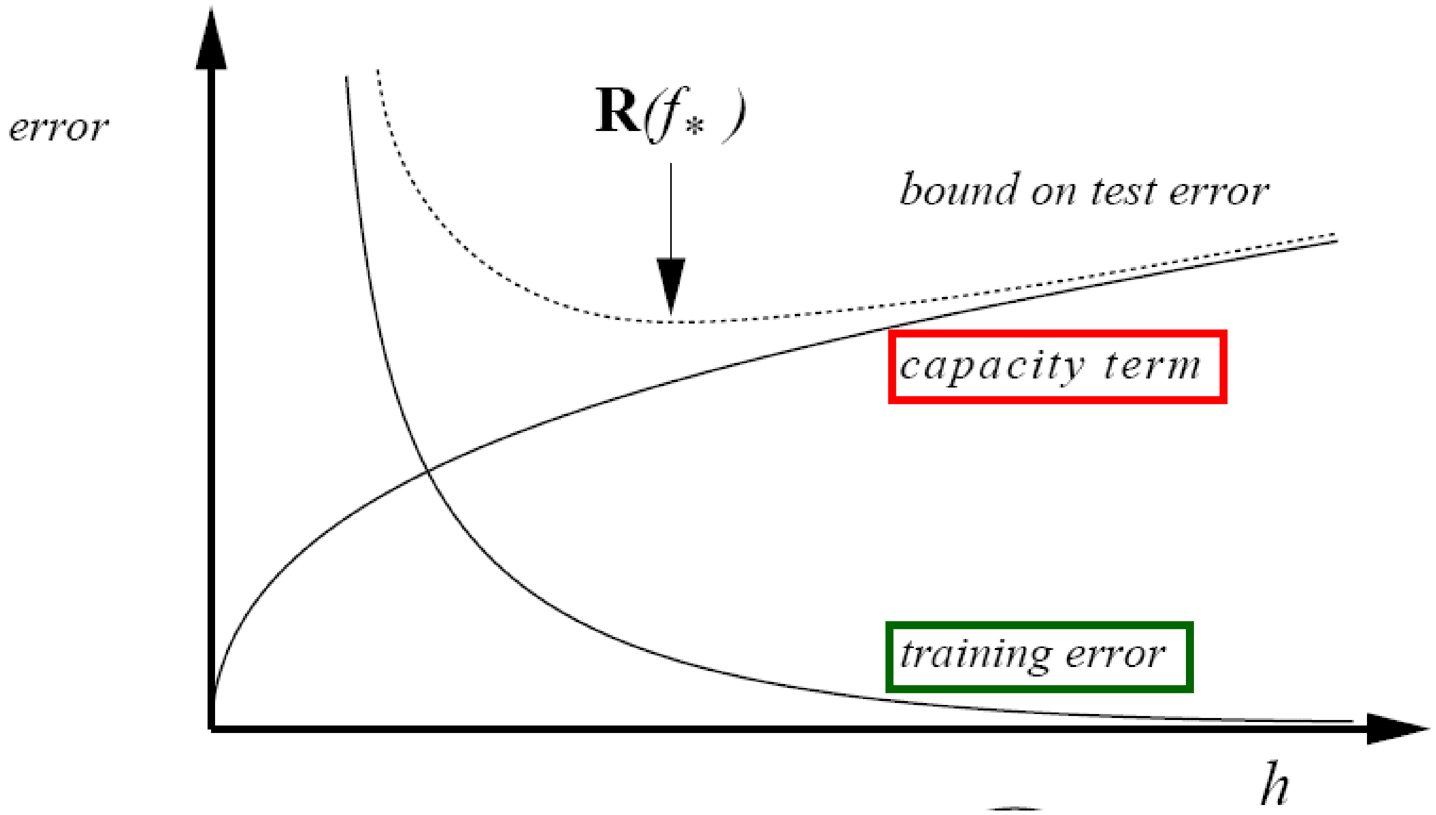
- Vapnik-Chervonenkis introduced a measure of the flexibility (**capacity**) of a classifier, the **VC-dimension**
- VC-dimension is the maximal number of samples the classifier can always classify correctly without any error
- For example a linear classifier in a 2D space has VC-dimension, h=3



Cannot be correctly classified!

Structural Risk Minimization

$$R[f] \leq \boxed{R_{emp}[f]} + \boxed{\phi\left(\frac{h}{n}\right)}$$



What about SVMs ?

- Recall that a linear separator in R^2 has VC-dimension, h=3
- In general a linear separator in R^N has VC-dimension, h =N+1
- A separating hyperplane in an high dimensional features space used can have a very high VC-dimension and thus generalizing bad
- Instead, large margin hyperplane <**w**,b> have a bounded VC-dimension:

$$h \cdot \quad R^2 jjwjj^2$$

- where R is the radius of the smallest sphere around the origin containing the training samples

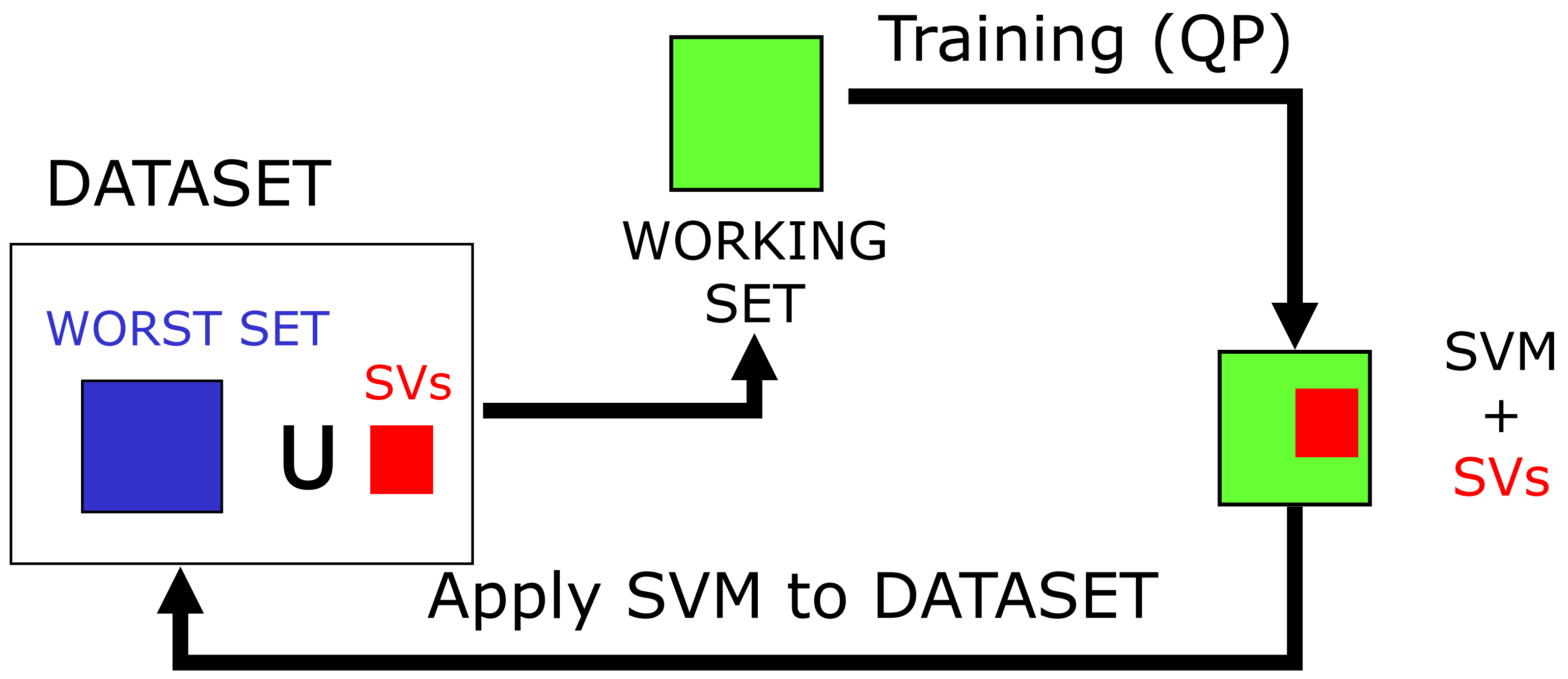
Part 4: Advanced Topics

- Training SVM
- Multi-class SVM
- Regression

SVM Training

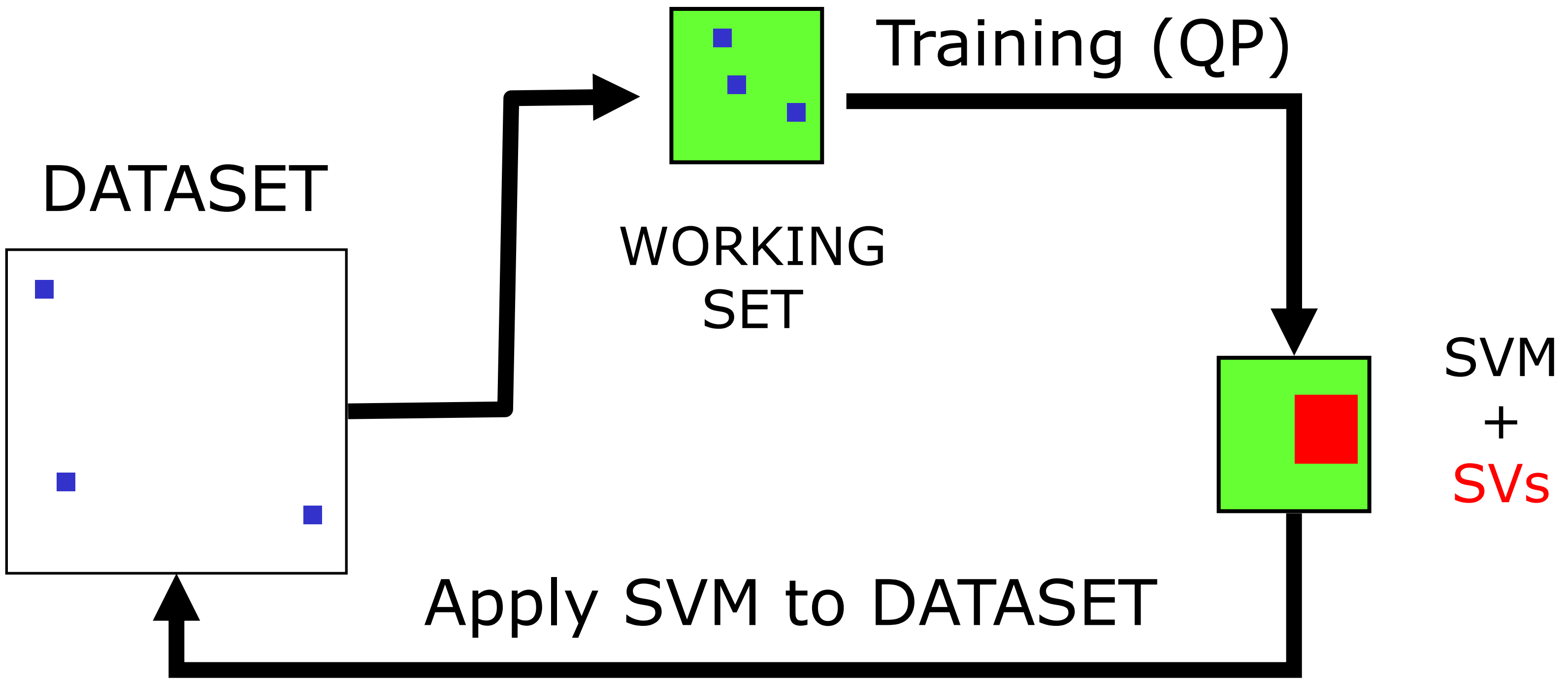
- Just solve the optimization problem to find α_i and b
- … but it is very expensive: $O(n^3)$ where n is the size of training set
- Faster approaches:
  - Chunking
  - Osuna’s methods
  - Sequential Minimal Optimization
- Online learning:
  - Chunking-based methods
  - Incremental methods

Chunking



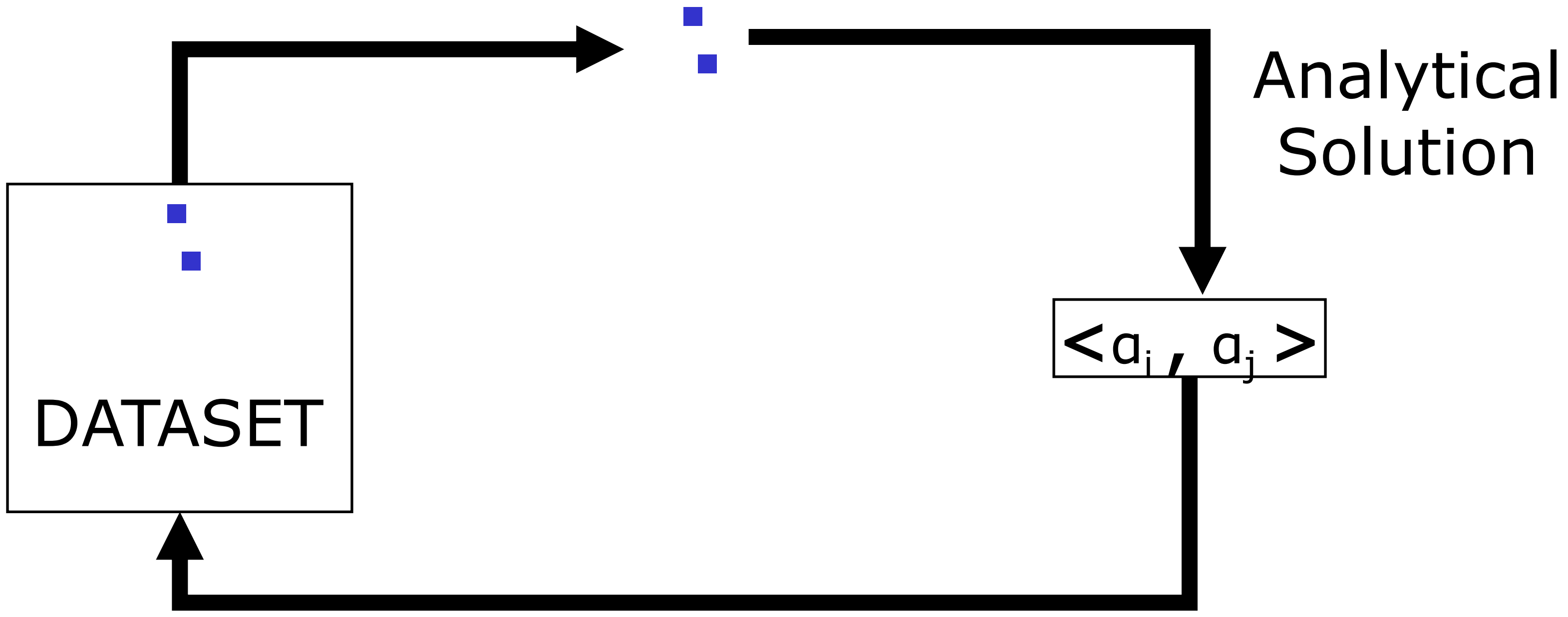
- Solves iteratively a sub-problem (working set)
- Build the working set with current SVs and the M samples with the bigger error (worst set)
- Size of the working set may increase!
- Converges to optimal solution!

Osuna’s Method



- Solves iteratively a sub-problem (working set)
- Replace some samples in the working set with missclassified samples in data set
- Size of the working set is fixed!
- Converges to optimal solution!

Sequential Minimal Optimization



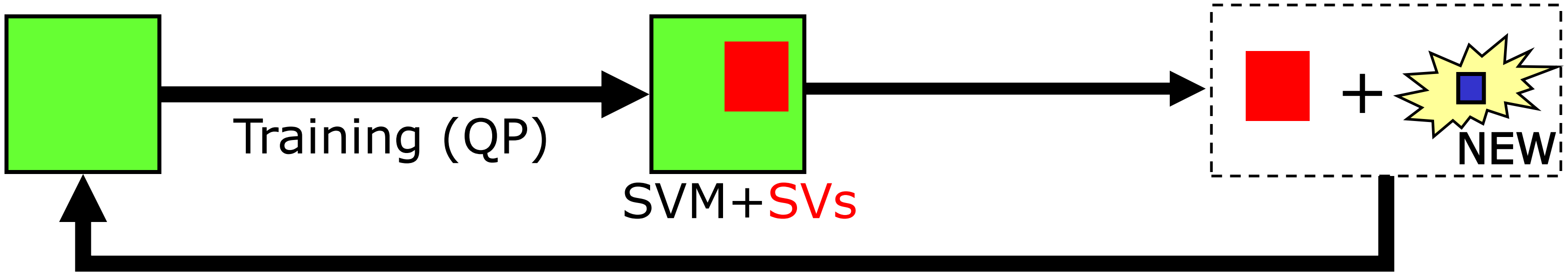
- Works iteratively only on two samples
- Size of the working set is minimal and multipliers are found analytically
- Converges to optimal solution!

Online Learning

- Batch Learning
  - All the samples known in advance
  - Training at once
- Online Learning
  - One sample at each time step
  - Adaptive Training
- Applications
  - Huge Dataset
  - Data streaming (e.g. satellite, financial markets)
- Approaches
  - Chunking-based
  - Incremental

Online Learning (2)

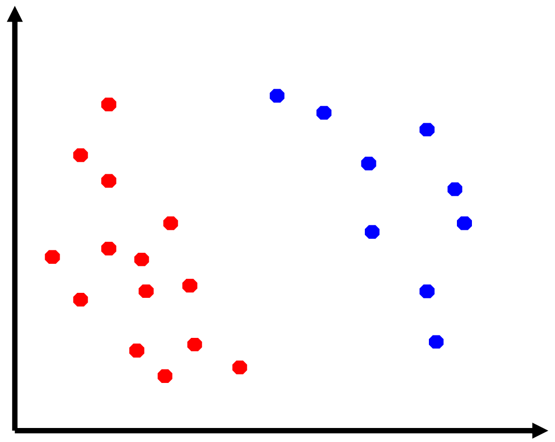
- Chunking-based methods



- Incremental methods
  - Adapting online the Lagrangian multipliers on the arrival of new samples
  - Recursive solution to the QP optimization problem
- Incremental methods are not widely used because they are too tricky and complex to implement

Multi-class SVM

- So far we considered two-classes problems:



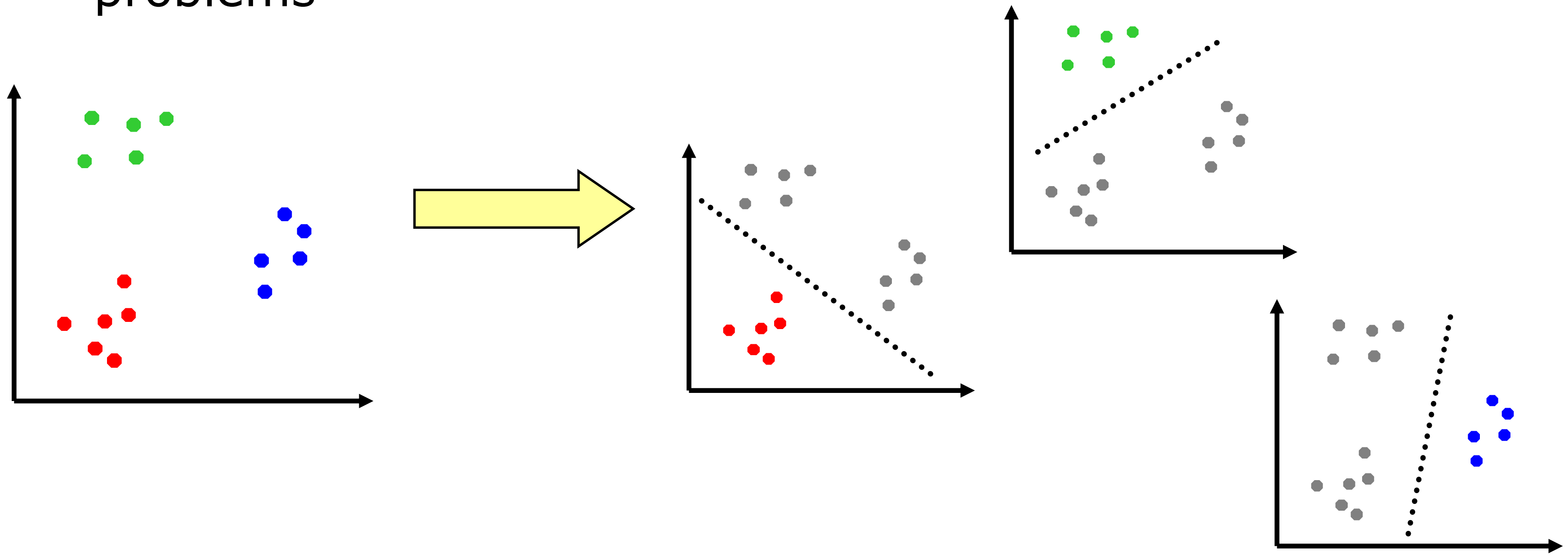
- How does SVM deal with multi-class problems?

Multi-class SVM: the approaches

- Adapt the problem formulation to a multi-class problems
  - Complex
  - Poor performing
- Reduce a k-class problem to N of 2-class problems
  - Computationally expensive
  - Simple (based on usual SVMs)
  - Good performance
- The second solution is widely used and we focus on it

One-against-all

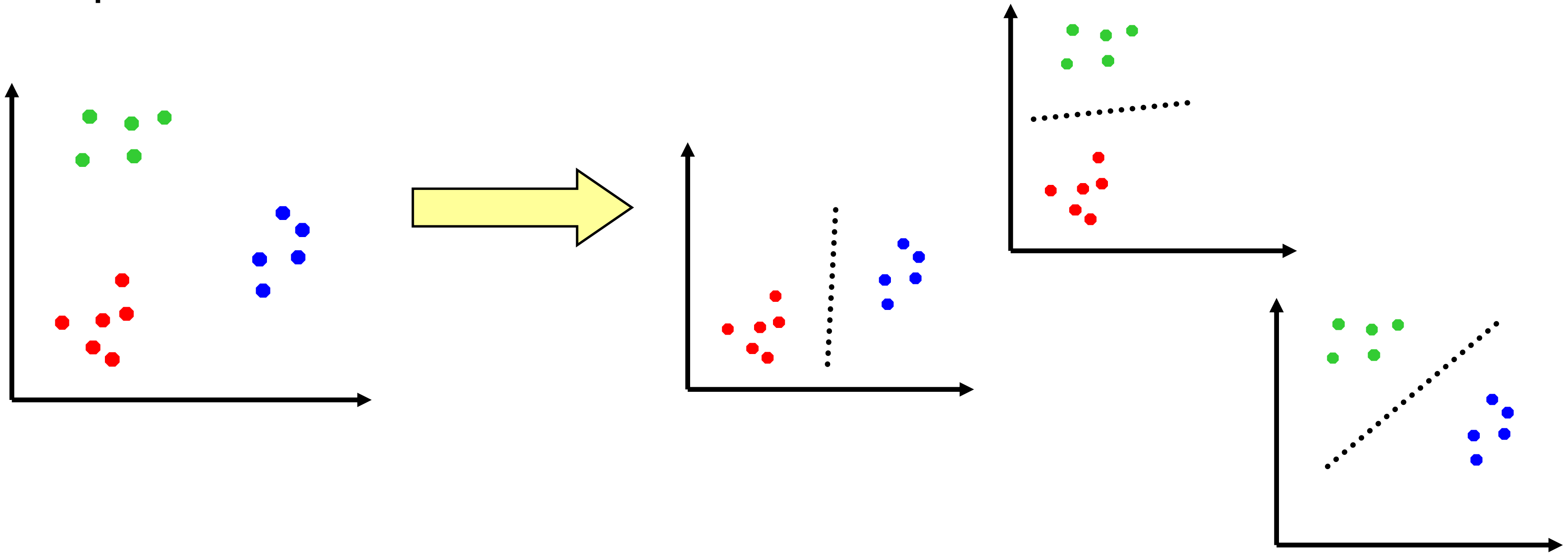
- A k-class problem is decomposed in k binary (2-class) problems



- Training is performed on the **entire dataset** and involves **k SVM classifiers**
- Test is performed choosing the class selected with the **highest margin** among the **k** SVM classifiers

One-against-one

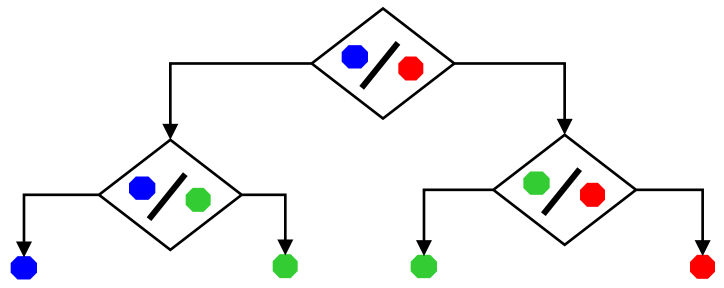
- A k-class problem is decomposed in k(k-1)/2 binary (2-class) problems



- The **k(k-1)/2 SVM classifiers** are trained on **subsets of the dataset**
- Test is performed by applying all the **k(k-1)/2** classifiers to the new sample and the **most voted** label is chosen

DAGSVM

- In DAGSVM, the k-class problem is decomposed in k(k-1)/2 binary (2-class) problems as in one-against-one
- Training is performed as in one-against-one
- But test is performed using a Direct Acyclic Graph to reduce the number of SVM classifiers to apply:

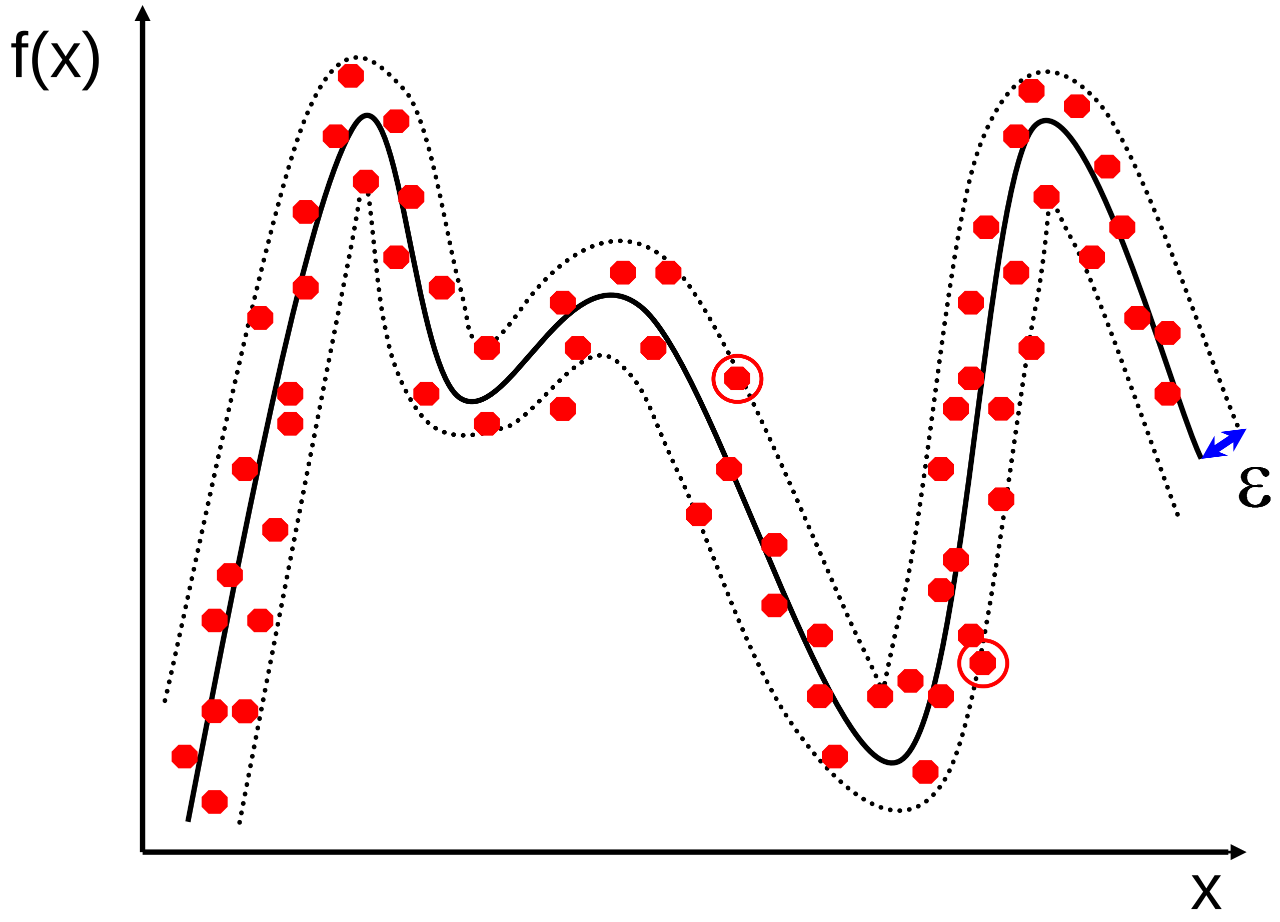


- The test process involves only **k-1** binary SVM classifiers instead of k(k-1)/2 as in one-against-one approach

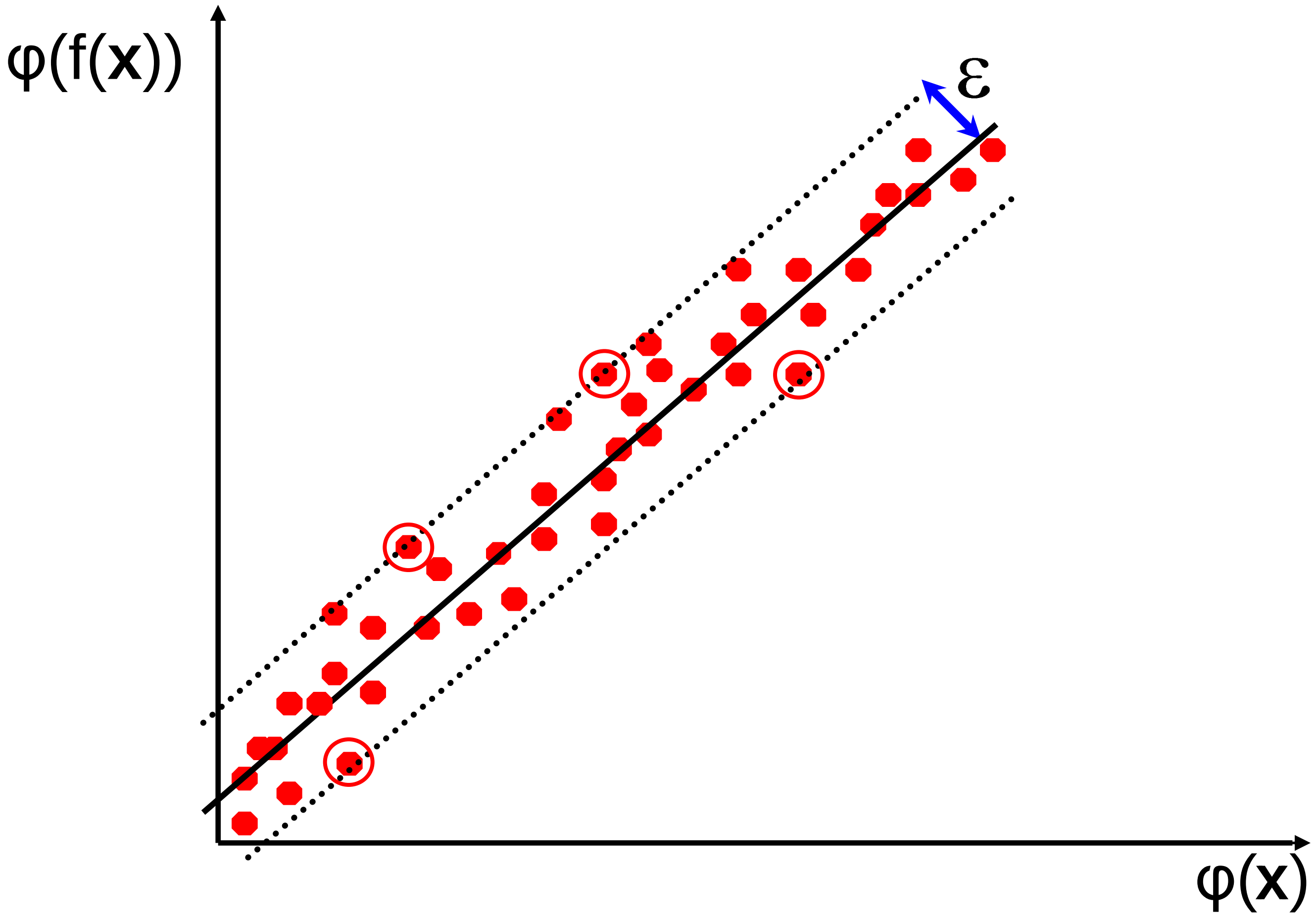
Multi-class SVM: summary

- One-against-all
  - **cheap** in terms of **memory** requirements
  - **expensive training** but **cheap test**
- One-against-one
  - **expensive** in terms of **memory** requirements
  - **expensive test** but **slightly cheap training**
- DAGSVM
  - **expensive** in terms of **memory** requirements
  - **slightly cheap training** and **cheap test**
- **One-against-one** is the **best performing** approach, due to the most effective decomposition
- **DAGSVM** is a **faster approximation** of one-against-one

Regression



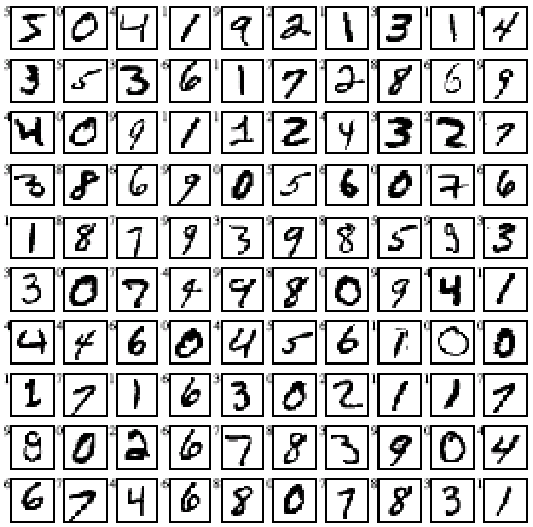
Regression



Part 5: Applications

- Handwriting recognition
- Face detection
- Learning to Drive

Handwriting recognition

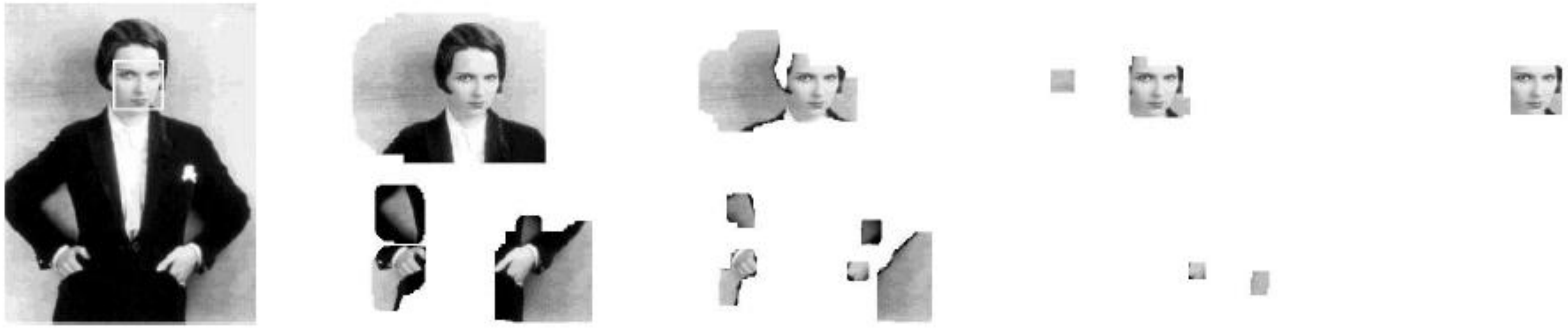


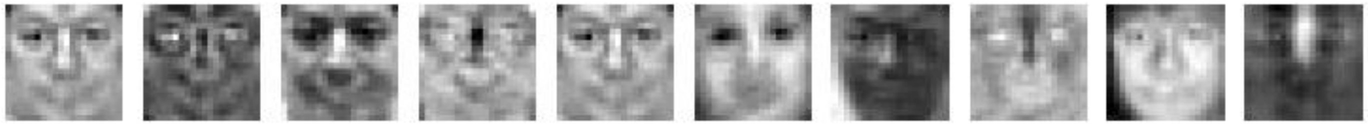
MNIST Benchmark

- 60000 training samples
- 10000 test samples
- 28x28 pixel

Classifier	Test Error
Linear Classifier	8.4%
3-nearest-neighbour	2.4%
SVM	1.4%
LeNet4	1.1%
Boosted LeNet4	0.7%
Translation Invariant SVM	0.56%

Face Detection

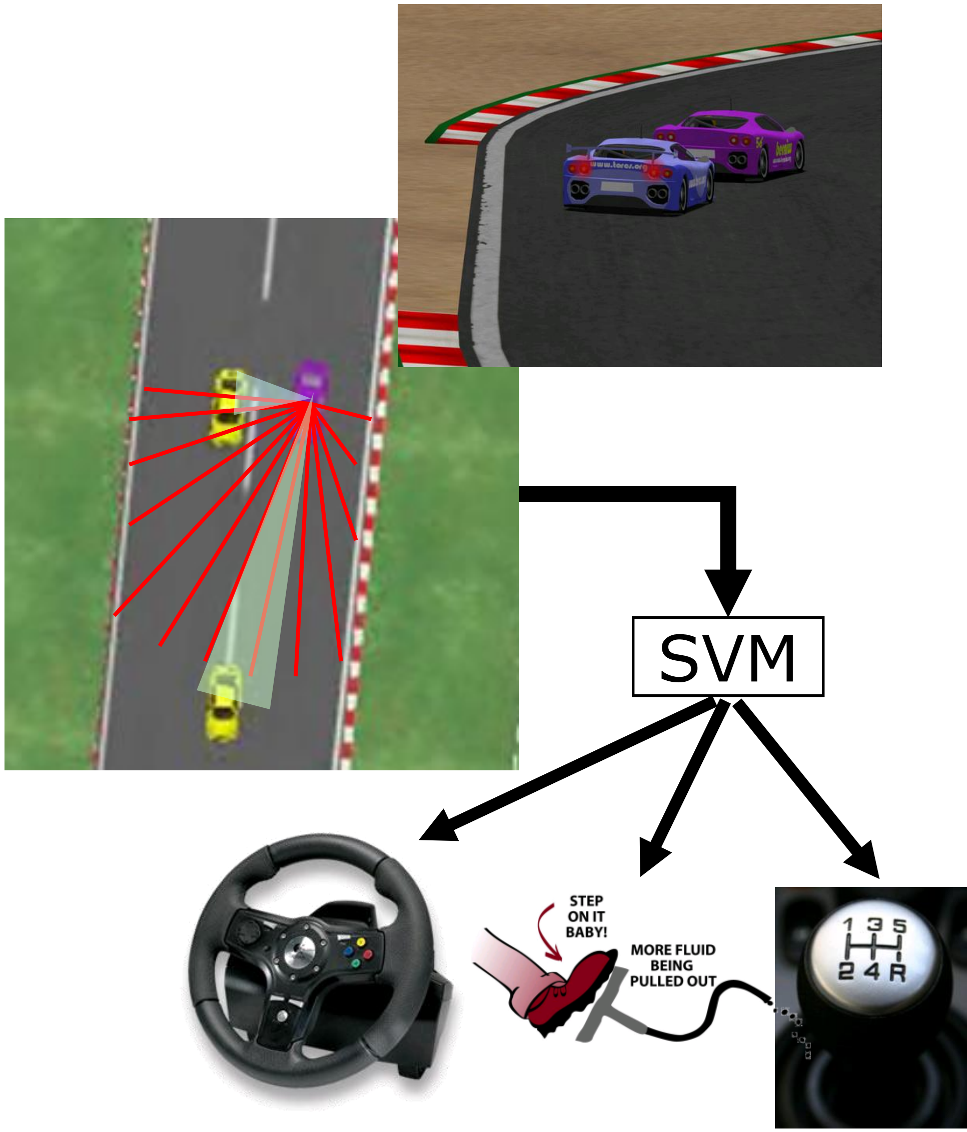




Templates

Learn to drive

- ❑ TORCS is an open source racing game
- ❑ Can a SVM learn to drive imitating a human driver?
- ❑ Player modeling has many interesting application in computer games:
  - human-like opponents
  - improve opponents prediction capabilities
  - automatic testing and design



Summary

- SVM search for an optimal **separating hyperplane**
- With **kernel trick**, SVM extend to **non-linear problems**
- SVM have a **strong theoretical background**
- SVM are **very expensive** to train
- SVM can be extended to
  - Multi-class problems
  - Regression problems
- SVM have been successfully applied to many problems

- Pointers
  - http://www.kernel-machines.org/
  - http://videolectures.net/
  - LIBSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)