Lecture 02: Kernel Classification and Clustering

Gustavo Camps-Valls

Image Processing Laboratory (IPL) – Universitat de Valncia. Spain.
gustavo.camps@uv.es, http://www.uv.es/gcamps

The organization of the course:

1. Fundamentals of kernel methods
2. Supervised and unsupervised kernel-based classification <<<
3. Kernel methods for regression and time series analysis
4. Nonlinear feature extraction with kernels

What's Classification?

"Classification refers to an algorithmic procedure for assigning a given piece of input data into one of a given number of categories."

What's Classification?

"Classification refers to an algorithmic procedure for assigning a given piece of input data into one of a given number of categories."

The classical approaches

The problem can be tackled with different levels of information:

- Supervised classification <<<<
- Unsupervised classification (clustering)
- SemiSupervised classification

What's Classification?

"Classification refers to an algorithmic procedure for assigning a given piece of input data into one of a given number of categories."

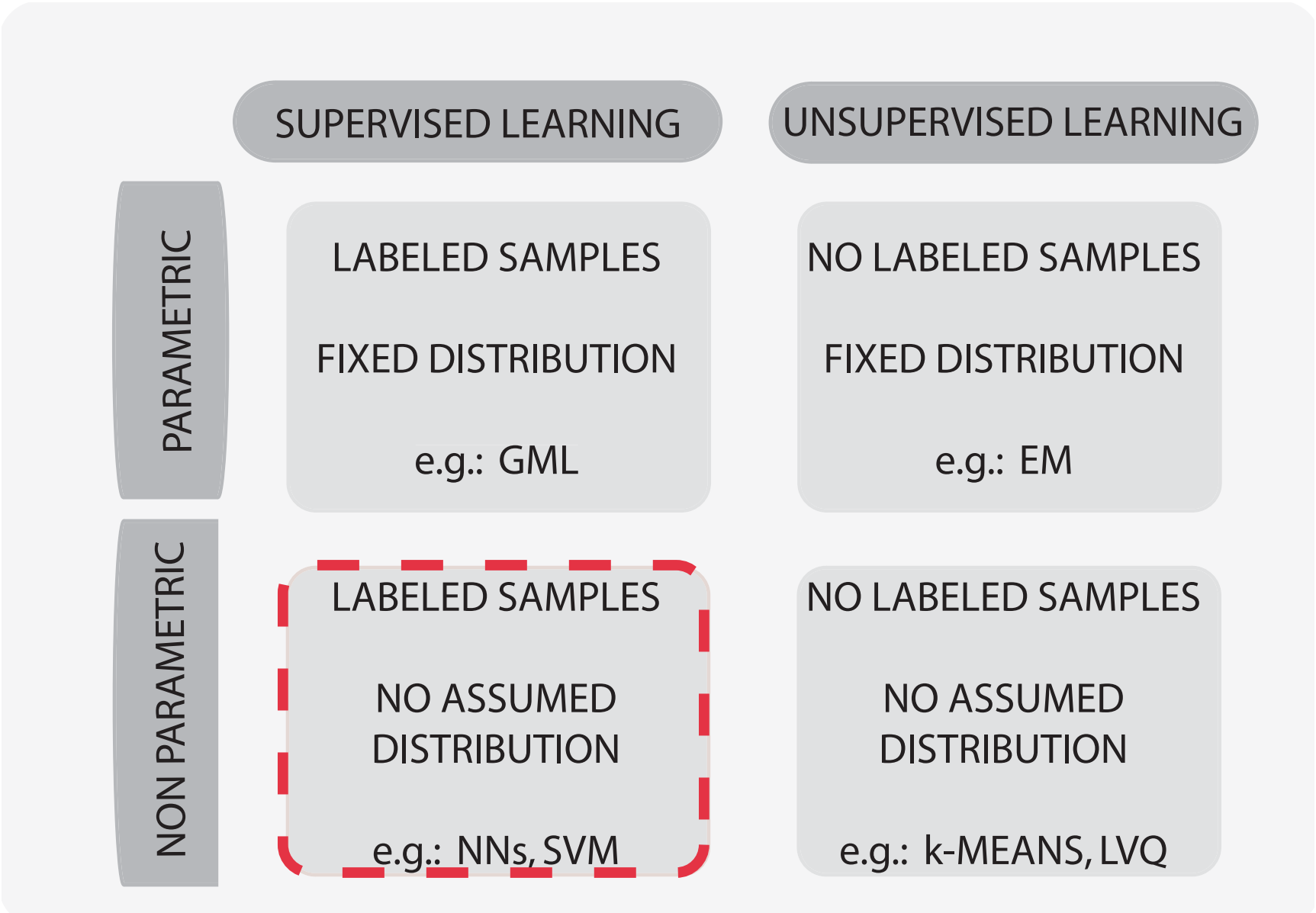
The classical approaches

The problem can be tackled with different levels of information:

- Supervised classification <<<<
- Unsupervised classification (clustering)
- SemiSupervised classification

Supervised Classification, formally?

Given pairs of input-output data $\{\mathbf{x}_i, \mathbf{y}_i\}$, $i = 1, \ldots, n$, learn a function that assigns a predicted label to new incoming samples $\mathbf{x}_*$, $y_* = f(\mathbf{x}_*)$.
[The function should be smooth (close samples should receive similar labels) and not too complex (parsimonious, regularized).]


SUPERVISED LEARNING
UNSUPERVISED LEARNING
PARAMETRIC
LABELED SAMPLES
FIXED DISTRIBUTION
e.g.: GML
NO LABELED SAMPLES
FIXED DISTRIBUTION
e.g.: EM
NON PARAMETRIC
LABELED SAMPLES
NO ASSUMED DISTRIBUTION
e.g.: NNs, SVM
NO LABELED SAMPLES
NO ASSUMED DISTRIBUTION
e.g.: k-MEANS, LVQ

Supervised Classification

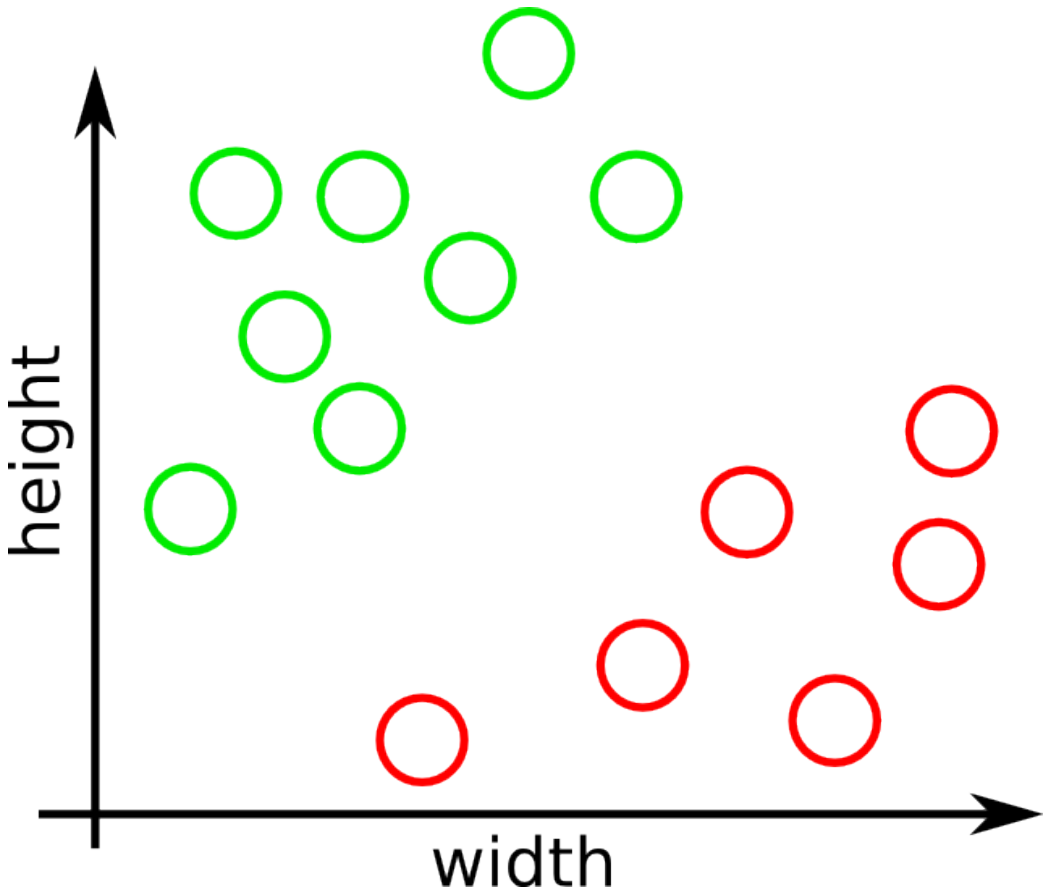
Many available classifiers:

1. Parallelepiped
2. Minimum distance
3. Linear/Quadratic Discriminant Analysis
4. Fuzzy models
5. Artificial Neural networks
6. Bayesian networks
7. ....

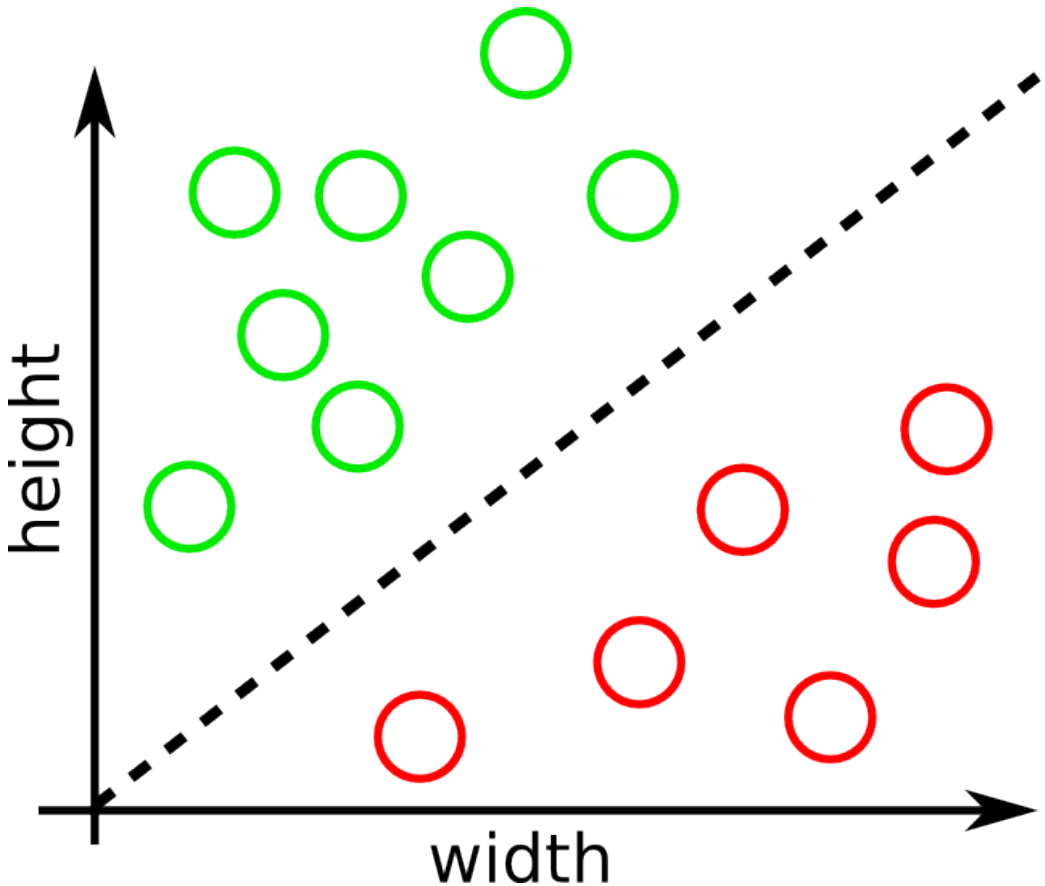
FORGET IT! GOING LINEAR IS COOL!

(if done in a proper feature space)

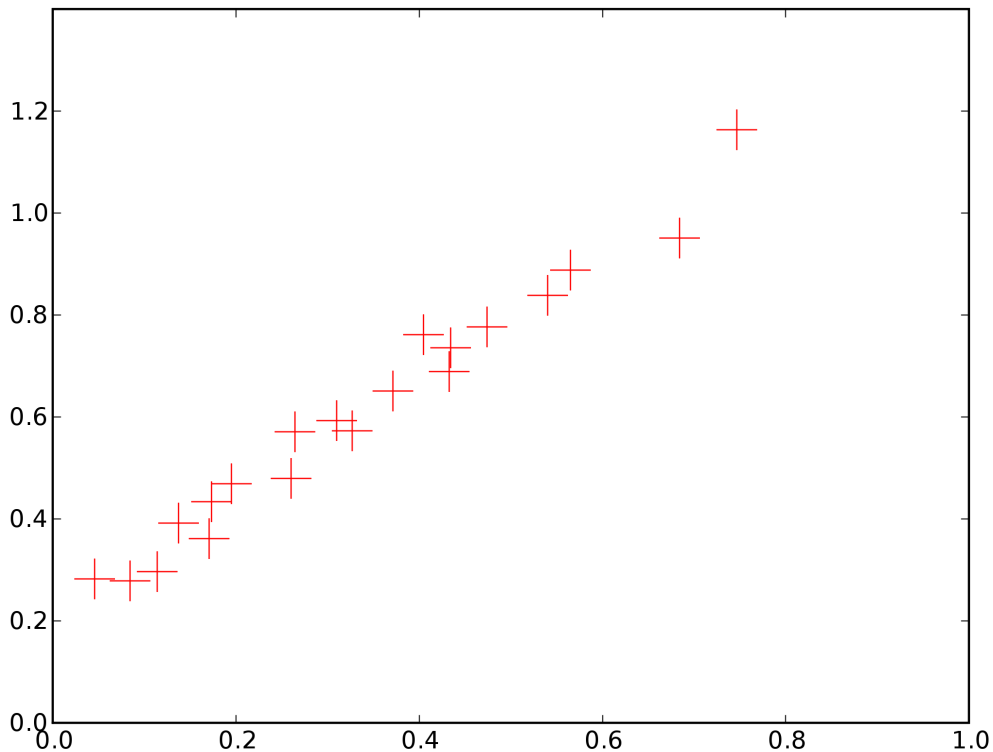
Find a hyperplane that separates between two sample sets.



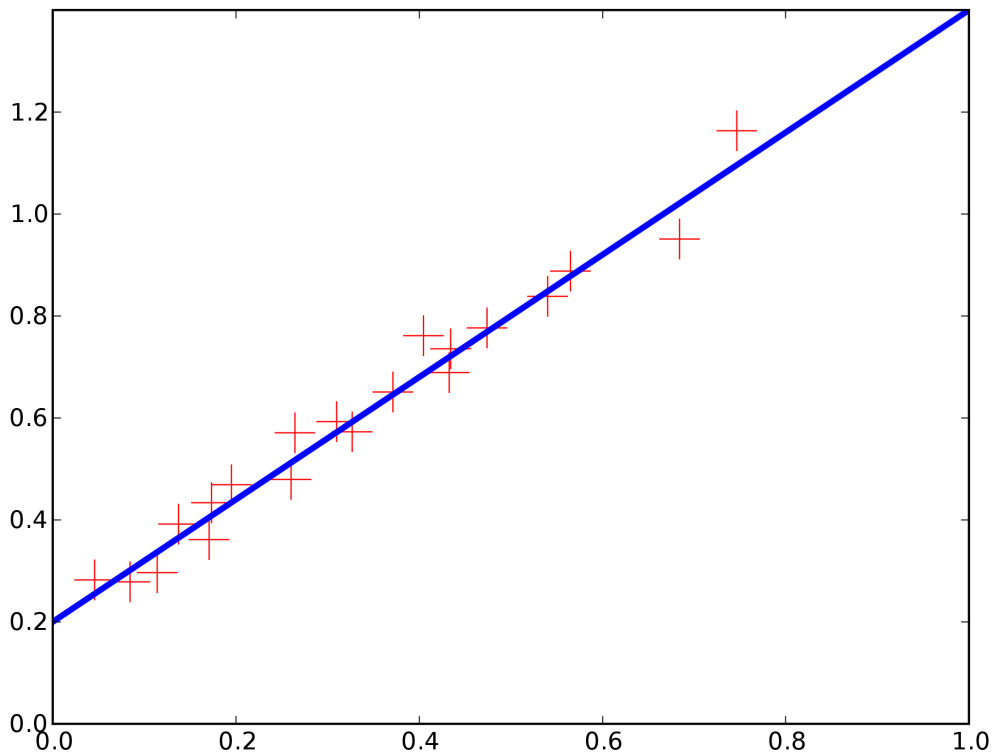
Find a hyperplane that separates between two sample sets.



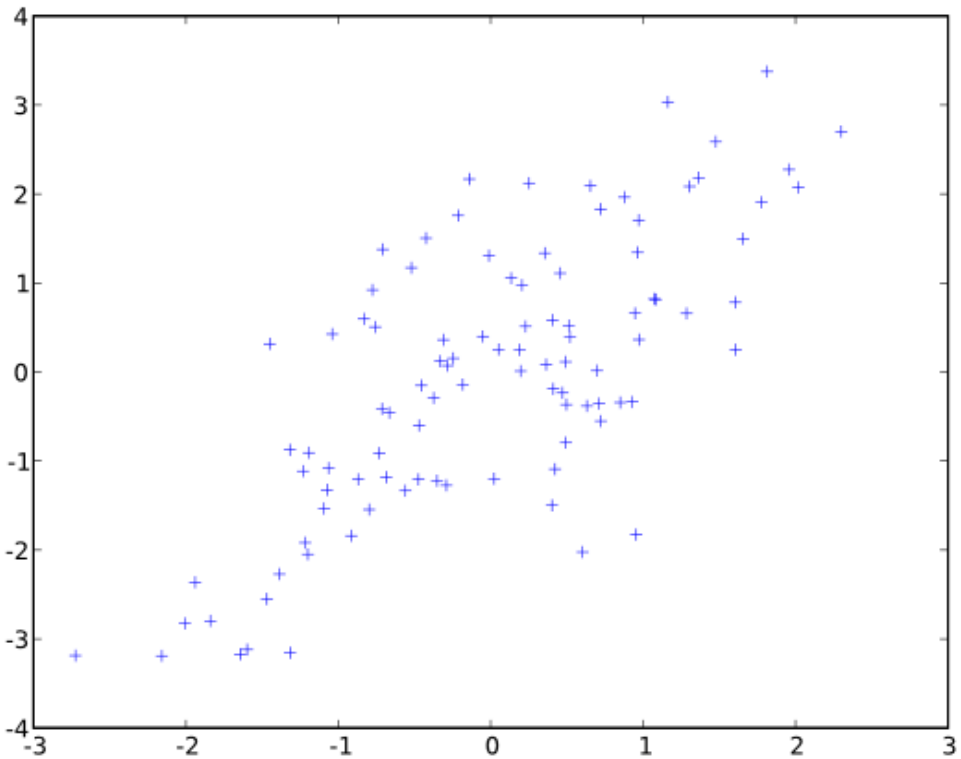
Find a linear function that interpolates data points.



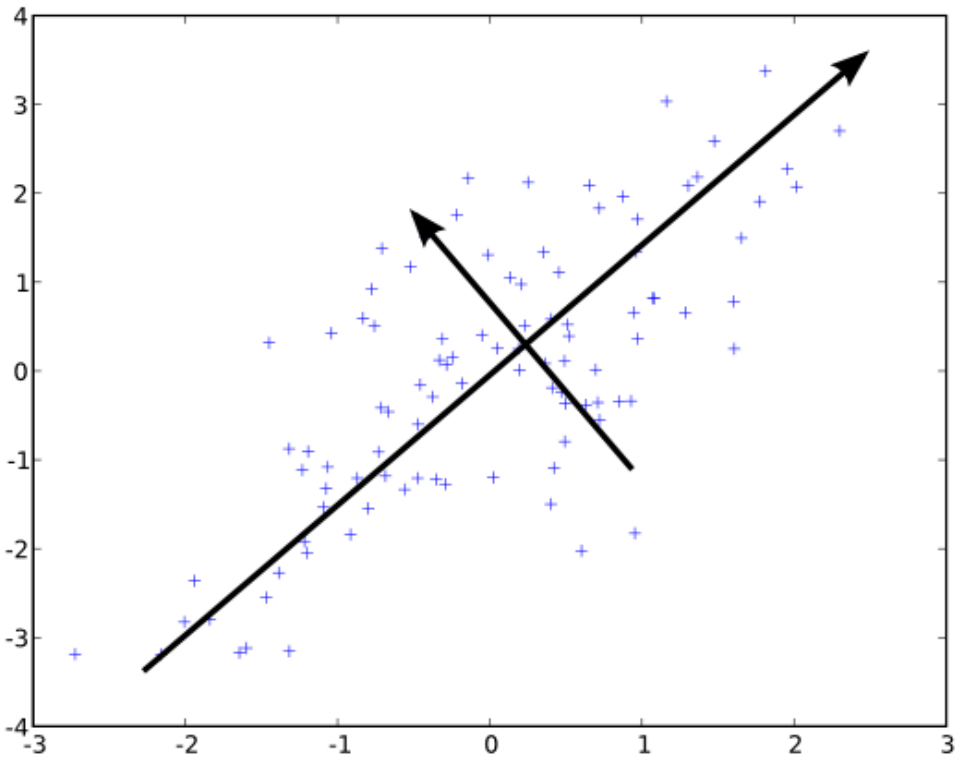
Find a linear function that interpolates data points.



Find a linear projections that preserve structure in the data.



Find a linear projections that preserve structure in the data.



Three different elementary tasks:

- classification,
- regression,
- dimensionality reduction.

In each case, linear techniques are very successful.

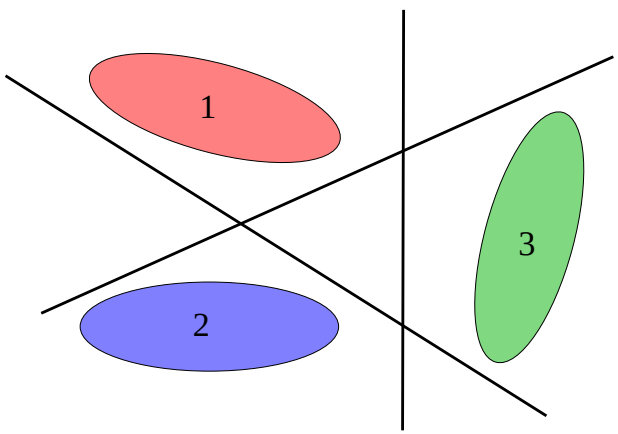
Why?

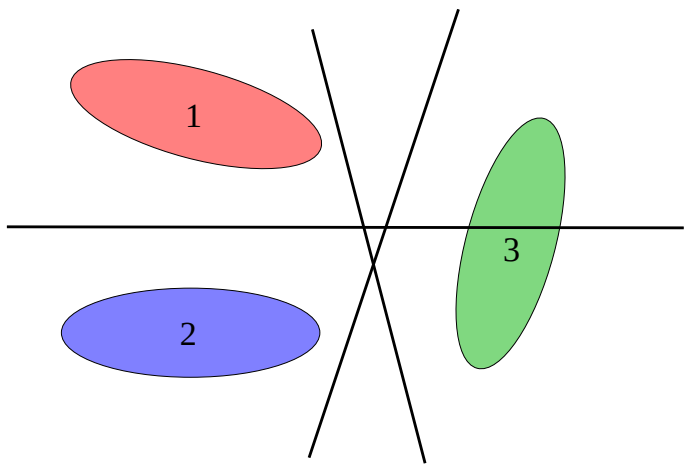
Linear techniques

- often work well,
  - most natural functions are smooth,
  - smooth function can locally be approximated by linear functions.
- are fast and easy to solve
  - elementary maths, even closed form solutions
  - typically involve only matrix operation
- are intuitive
  - solution can be visualized geometrically,
  - solution corresponds to common sense.

With several classes, two common strategies:

- one-vs-all, one-vs-the-rest: c classifiers
- one-vs-one, $c(c-1)/2$ classifiers





Regularized least squares linear classification

- Inputs: $\mathbf{X} \in \mathbb{R}^{n \times d}$
- Outputs: $\mathbf{Y} \in \mathbb{R}^{n \times 1}$, $\mathbf{Y} = [y_1, y_2, \ldots, y_n]^\top$
- Model: $\mathbf{Y} = sign(\mathbf{X}\mathbf{w})$
- Functional:

$$\mathbf{w}^* = \min_{\mathbf{w}} \left\{ \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2 \right\}$$

- After deriving and setting to zero, $\mathbf{w} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_d)^{-1} \mathbf{X}^\top \mathbf{Y}$

Regularized least squares linear classification

- Inputs: $\mathbf{X} \in \mathbb{R}^{n \times d}$
- Outputs: $\mathbf{Y} \in \mathbb{R}^{n \times 1}$, $\mathbf{Y} = [y_1, y_2, \ldots, y_n]^\top$
- Model: $\mathbf{Y} = sign(\mathbf{X}\mathbf{w})$
- Functional:

$$\mathbf{w}^* = \min_{\mathbf{w}} \left\{ \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2 \right\}$$

- After deriving and setting to zero, $\mathbf{w} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_d)^{-1} \mathbf{X}^\top \mathbf{Y}$

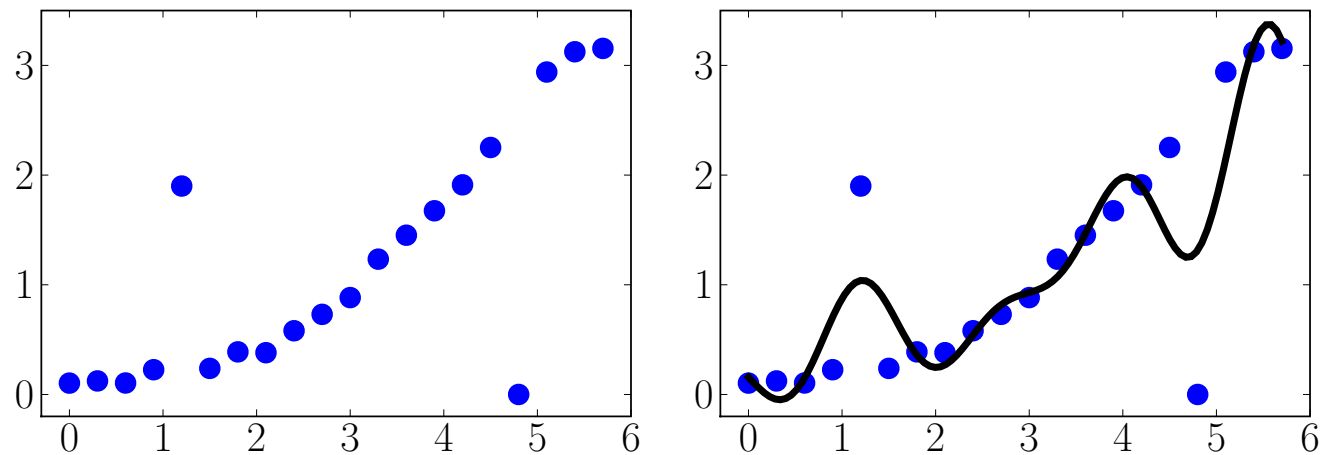
Regularized kernel least squares classification

- Model: $\mathbf{Y} = sign(\mathbf{\Phi}\mathbf{w}_\mathcal{H})$
- Functional:

$$\mathbf{w}^*_\mathcal{H} = \min_{\mathbf{w}_\mathcal{H}} \left\{ \|\mathbf{Y} - \mathbf{\Phi}\mathbf{w}_\mathcal{H}\|^2 + \lambda \|\mathbf{w}_\mathcal{H}\|^2 \right\}$$

- Dual weights: $\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} \mathbf{Y}$
- Primal weigths: $\mathbf{w}_\mathcal{H} = \mathbf{\Phi}^\top \boldsymbol{\alpha}$
- Decision function $\mathbf{Y} = sign(\mathbf{\Phi}\mathbf{w}_\mathcal{H}) = sign(\mathbf{K}\boldsymbol{\alpha})$

Like Least-Squared Regression, (Kernel) Ridge Regression is sensitive to outliers:



because the quadratic loss function penalized large residue.

Problems!

- One weight per example $\longrightarrow$ Risk of overfitting
- High computational cost for $n > 2000$, different (σ, λ) to try!

Problems!

- One weight per example $\rightarrow$ Risk of overfitting
- High computational cost for $n > 2000$, different (σ, λ) to try!

Solutions

- Do proper cross validation and control λ. Plot λ- RMSE_{test} !!!
- Standard code: `alpha = inv(gamma + K) * Y;`
- Cholesky decomposition is faster ($\sim$4-fold) but not sparse again:
  ```
  R = chol(K+gamma*eye(n));
  alpha = R\(R'\Y);
  ```
- Nyström method uses the Sherman-Morrison-Woodbury formula:

$$(\mathbf{A} + \mathbf{V}\mathbf{V}^\top)^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{V}(\mathbf{I} + \mathbf{V}^\top\mathbf{D}^{-1}\mathbf{V})^{-1}\mathbf{V}^\top\mathbf{D}^{-1}$$

Problems!

- One weight per example $\rightarrow$ Risk of overfitting
- High computational cost for $n > 2000$, different (σ, λ) to try!

Solutions

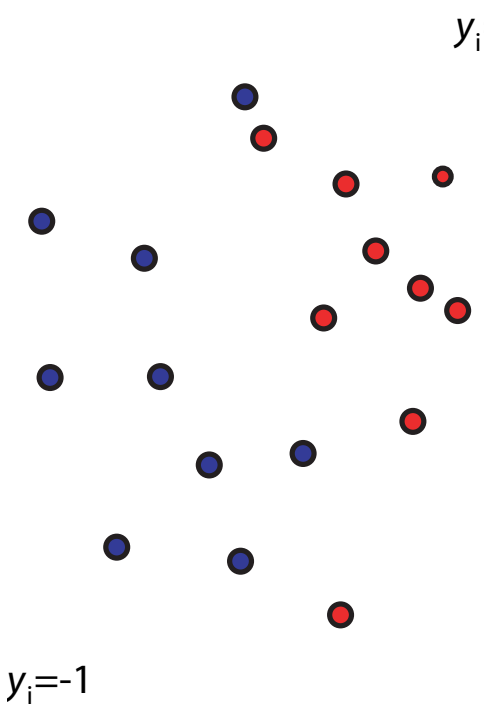
- Do proper cross validation and control λ. Plot λ- RMSE_{test} !!!
- Standard code: `alpha = inv(gamma + K) * Y;`
- Cholesky decomposition is faster ($\sim$4-fold) but not sparse again:
  ```
  R = chol(K+gamma*eye(n));
  alpha = R\(R'\Y);
  ```
- Nyström method uses the Sherman-Morrison-Woodbury formula:

$$(\mathbf{A} + \mathbf{V}\mathbf{V}^\top)^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{V}(\mathbf{I} + \mathbf{V}^\top\mathbf{D}^{-1}\mathbf{V})^{-1}\mathbf{V}^\top\mathbf{D}^{-1}$$

- There are tricks to make the KLSC sparse (not very naturally)

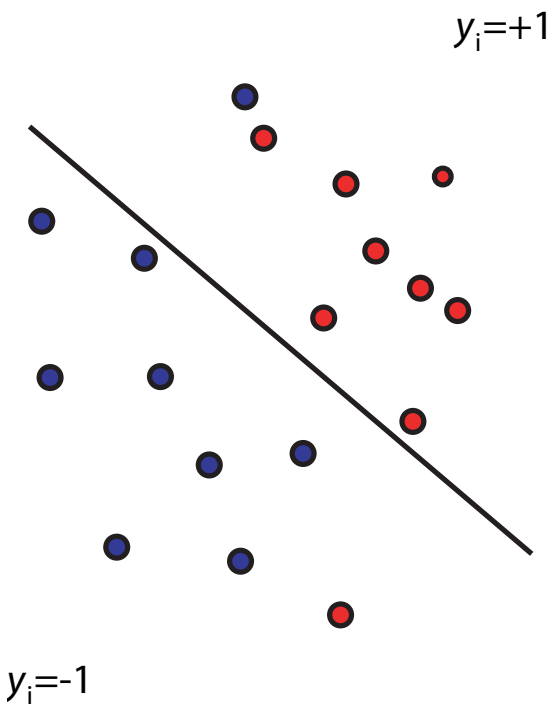
SVM linear classification

- **Data:** Given n examples $\mathbf{x}_i \in \mathbb{R}^N$ and $y_i \in \{-1, +1\}$ (classes)
- **Objective:** Build a linear classifier, $\hat{y} = f(\mathbf{x}) = sign(\mathbf{w}^\top \mathbf{x} + b)$.



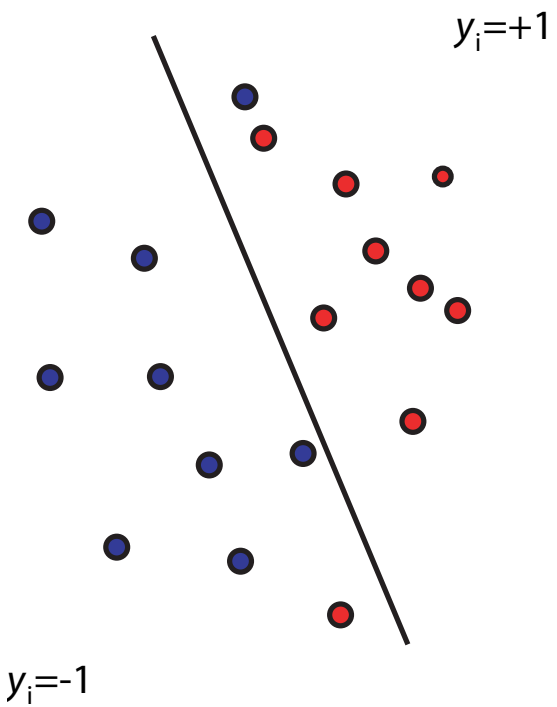
SVM linear classification

- **Several solutions exist!**
- **Objective:** Define the optimal one $(\mathbf{w}, b)$



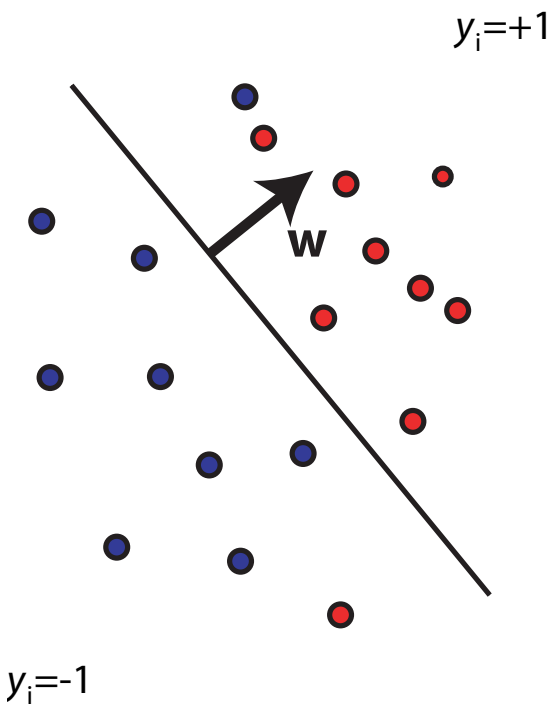
SVM linear classification

- **Several solutions exist!**
- **Objective:** Define the optimal one $(\mathbf{w}, b)$



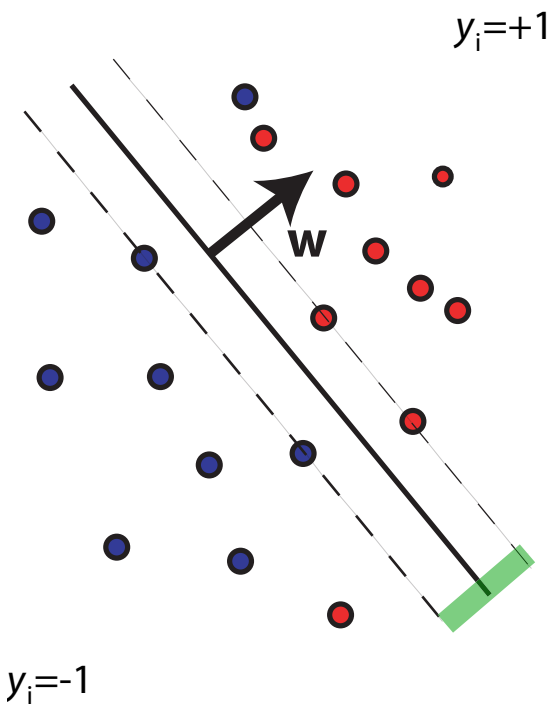
SVM linear classification

- **Intuitively there's an optimal one!**
- **Objective:** Define the optimal one $(\mathbf{w}, b)$



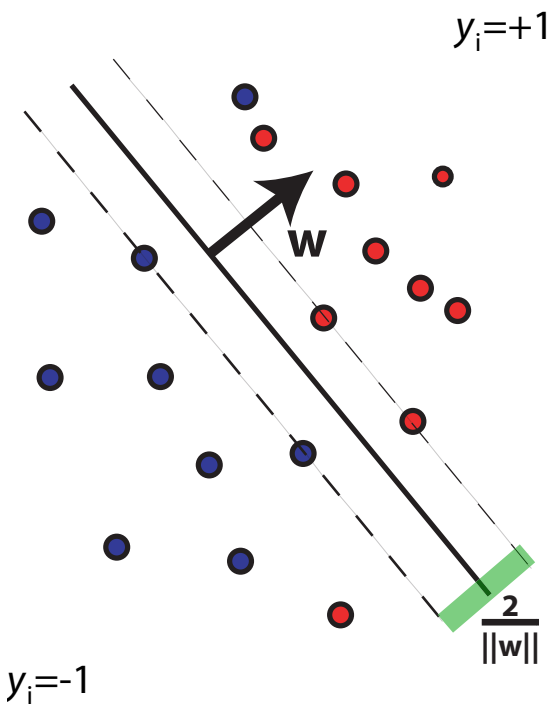
SVM linear classification

- **... and should separate samples from different classes maximally!**
- **Objective:** Define the optimal one $(\mathbf{w}, b)$



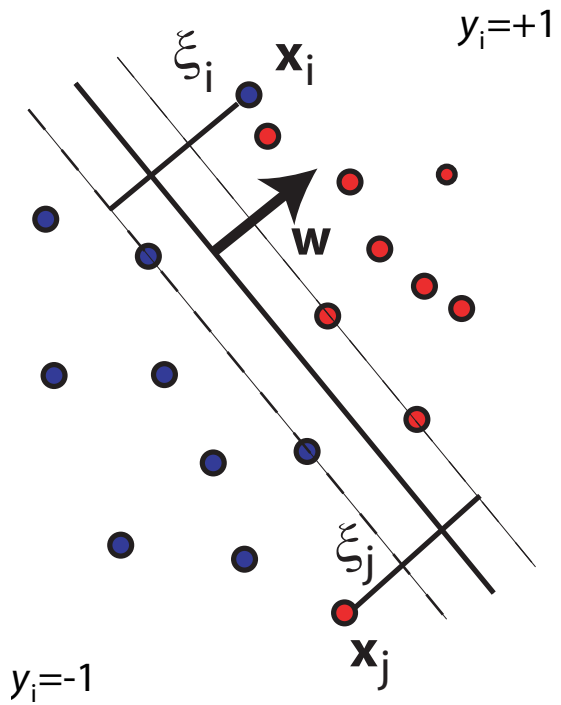
SVM linear classification

- **Maximize margin separation = minimize $\|\mathbf{w}\|$:** $\min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 \right\}$



SVM linear classification

- **Errors must be also penalized!** $\min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_i \xi_i \right\}$



SVM linear classification

- **... and examples are forced to belong to their class!**

$$\min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \right\}$$

subject to:

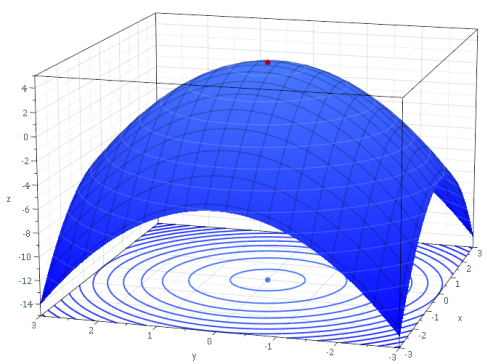
$$\begin{aligned} \mathbf{w}^\top \mathbf{x}_i + b &\geq 1 - \xi_i \quad y_i = +1, \forall i = 1, ..., n \\ \mathbf{w}^\top \mathbf{x}_i + b &\leq 1 - \xi_i \quad y_i = -1, \forall i = 1, ..., n \\ \xi_i &\geq 0 \end{aligned}$$

- ξ_i is the error associated to misclassify example $\mathbf{x}_i$
- C is a tradeoff parameter controlling the overfitting

A brief review of optimization techniques

Optimization

- Optimization means solving problems by minimizing (or maximizing) a real function by choosing the values of real or integer variables from an allowed set.
- Many methods and families of techiques:
  1. Gradient descent (first-order optimization): takes steps proportional to the negative of the gradient of the function at the current point
  2. Linear programming: if you have a linear objective function with constraints
  3. Quadratic programming: if you have a squared objective function with constraints
  4. Semidefinite programming
  5. Convex cone optimization
  6. ....



A brief review of optimization techniques

Linear programming

"Linear programming (LP) determines the solution of a linear objective function, subject to linear equality and linear inequality constraints."

$$\max_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} \qquad s.t. \qquad \mathbf{Ax} \leq \mathbf{b}$$

where $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{A}$ are known.

Quadratic programming

"Quadratic programming (QP) determines the solution of a quadratic function subject to linear constraints."

$$\max_{\mathbf{x}} \frac{1}{2}\mathbf{x}^\top \mathbf{K}\mathbf{x} + \mathbf{c}^\top \mathbf{x} \qquad s.t. \qquad \mathbf{Ax} \leq \mathbf{b} \qquad and \qquad \mathbf{Ex} = \mathbf{d}$$

A brief review of optimization techniques

Linear programming

"Linear programming (LP) determines the solution of a linear objective function, subject to linear equality and linear inequality constraints."

$$\max_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} \qquad s.t. \qquad \mathbf{Ax} \leq \mathbf{b}$$

where $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{A}$ are known.

Quadratic programming

"Quadratic programming (QP) determines the solution of a quadratic function subject to linear constraints."

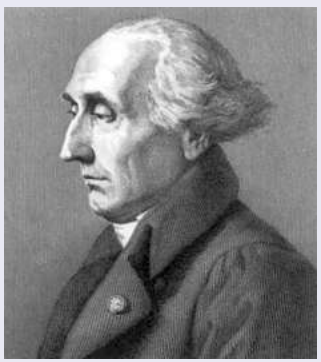
$$\max_{\mathbf{x}} \frac{1}{2}\mathbf{x}^\top \mathbf{K}\mathbf{x} + \mathbf{c}^\top \mathbf{x} \qquad s.t. \qquad \mathbf{Ax} \leq \mathbf{b} \qquad and \qquad \mathbf{Ex} = \mathbf{d}$$

Tons of efficient solvers

- Matlab's `quadprog`, `linprog`, `simplex`
- Open source C++ tools: QSOpt, SMO, Pegasus, `libLBFGS`

A brief review of optimization techniques

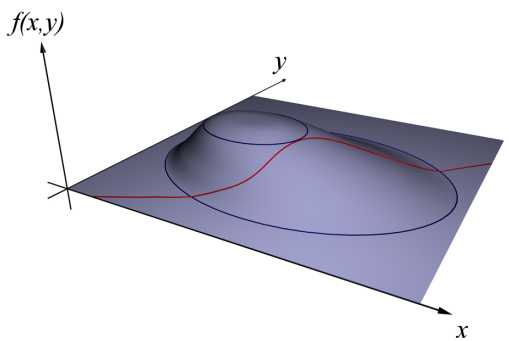
Lagrange



Joseph-Louis Lagrange (1713-1813), Giuseppe Lodovico (Luigi) Lagrangia: 'optimization of functions of several variables subject to equality and inequality constraints'

Method of Lagrange multipliers

$$\max_{x,y} f(x,y) \qquad s.t. \qquad g(x,y) = c$$



A brief review of optimization techniques

Method of Lagrange multipliers

$$\max_{x,y}\{f(x,y)\} \qquad s.t. \qquad g(x,y) \leq c$$

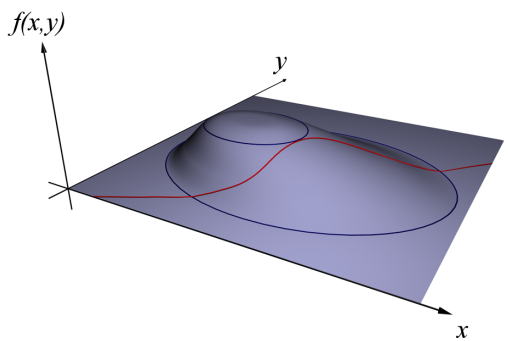
Method of Lagrange multipliers: solution

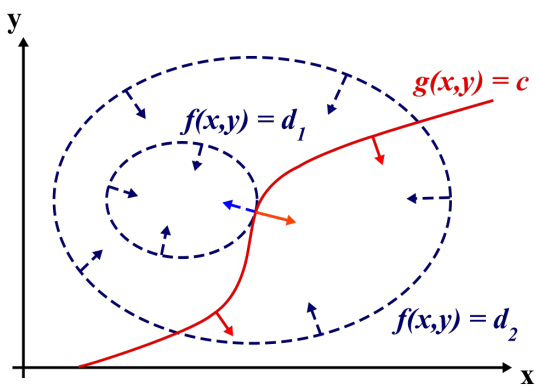
Introduce a new variable (named 'Lagrange multiplier' α) and optimize the new function:

$$\max_{x,y,\lambda}\{\Lambda(x,y,\alpha)\} \qquad s.t. \qquad f(x,y) - \alpha(g(x,y) - c)$$

which is equivalent to solve the dual problem

$$\max_{x,y,\lambda}\{f(x,y) - \alpha(g(x,y) - c)\} \qquad s.t. \qquad \alpha \geq 0$$





Method of Lagrange multipliers

- **Lagrange duality:** The method is applicable to any number of variables and constraints:

  $$\min\{f_o(x)\} \qquad s.t. f_i(x) \leq 0 \qquad \textit{and} \qquad h_j(x) = 0$$

  The dual problem reduces to minimize:

  $$\Lambda(x, \alpha, \mu) = f_o(x) - \sum_i \alpha_i f_i(x) - \sum_j \mu_j h_i(x)$$

- **The procedure is very simple:**
  1. Derive this primal-dual problem and equal to zero, $\nabla_\lambda \Lambda(x, y, \alpha) = 0$
  2. The obtained constraints are stationary points of the solution
  3. Include the obtained constraints again
  4. Done! You obtain an LP or QP problem, for which there are lots of efficient solvers

A brief review of optimization techniques

- **Remember the SVM primal problem:**

$$\min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \right\}$$

subject to:

$$\begin{aligned} \mathbf{w}^\top \mathbf{x}_i + b &\geq 1 - \xi_i \quad y_i = +1, \forall i = 1, ..., n \\ \mathbf{w}^\top \mathbf{x}_i + b &\leq 1 - \xi_i \quad y_i = -1, \forall i = 1, ..., n \\ \xi_i &\geq 0 \end{aligned}$$

- ξ_i is the error associated to misclassify example $\mathbf{x}_i$
- C is a tradeoff parameter controlling the overfitting

A brief review of optimization techniques

The latter is a quadratic programming problem with linear constraints:

- Use Lagrange multipliers ($\xi_i \geq 0, \mu_i \geq 0$) for each constraint:

$$\min_{\mathbf{w},\boldsymbol{\xi},b} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(\mathbf{w}^\top \mathbf{x}_i + b) + \xi_i - 1] - \sum_{i=1}^{n}\mu_i\xi_i \right\}$$

- Then derive and equal to zero: $\frac{\partial L}{\partial \mathbf{w}} = 0$, $\frac{\partial L}{\partial \boldsymbol{\xi}} = 0$, $\frac{\partial L}{\partial b} = 0$
- Combining the previous results, we obtain the equivalent dual problem:

$$\max_{\boldsymbol{\alpha}} \left\{ -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \underbrace{\mathbf{x}_i^\top \mathbf{x}_j}_{=K_{ij}} + \sum_{i=1}^{n}\alpha_i \right\}$$

- After some operations, the decision function is:

$$\hat{y}_j = f(\mathbf{x}_j) = sign(\mathbf{w}^\top \mathbf{x}_j + b) = sign\left(\sum_{i=1}^{n}\alpha_i y_i \underbrace{\mathbf{x}_i^\top \mathbf{x}_j}_{=K_{ij}} + b \right)$$

- **Both for training and prediction, we only need the dot product (similarities) between vectors, K_{ij}, not the examples**

SVM nonlinear classification

- **Step 1:** Map examples to a higher dimensional space, $\phi : \mathbb{R}^N \rightarrow \mathcal{H}$

$$\mathbf{x}_i \rightarrow \phi(\mathbf{x}_i), \qquad i = 1, \ldots, n$$

- **Step 2:** Replace dot products $\langle \cdot, \cdot \rangle$ in $\mathcal{H}$ by a kernel function $k(\cdot, \cdot)$

$$\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = K(\mathbf{x}_i, \mathbf{x}_j)$$

- **Step 3:** Solve the same maximum margin problem

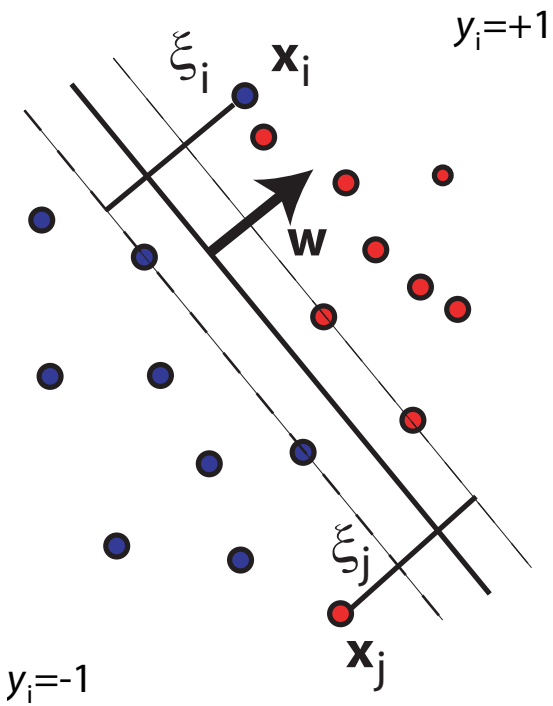
$$\max_{\boldsymbol{\alpha}} \left\{ -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \underbrace{\phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)}_{=K_{ij}} + \sum_{i=1}^{n} \alpha_i \right\}$$

- **Step 4:** Prediction involves comparing test to train samples with $K(\cdot, \cdot)$:

$$\hat{y}_j = f(\mathbf{x}_j) = sign(\mathbf{w}^\top \phi(\mathbf{x}_j) + b) = sign\left(\sum_{i=1}^{n} \alpha_i y_i \underbrace{\phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)}_{=K_{ij}} + b \right)$$

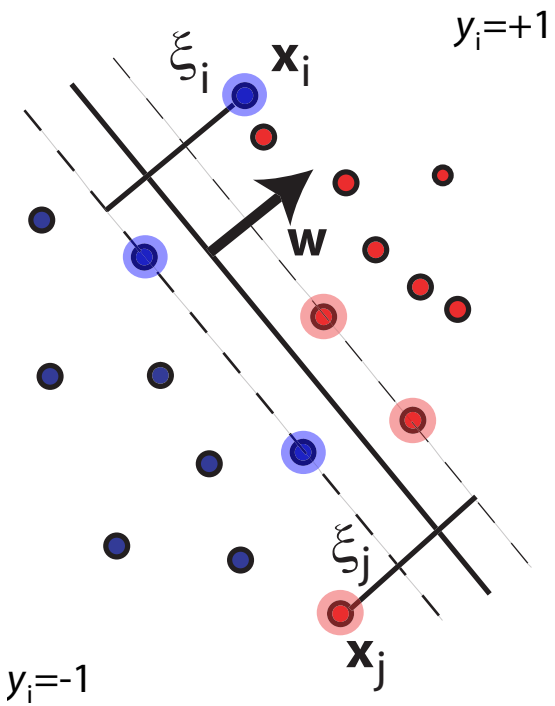
SVM nonlinear classification

- **The solution is sparse:** only few examples $\mathbf{x}_i$ with $\alpha_i \neq 0$ are important
- **Support vectors:** define the margin and are misclassified examples



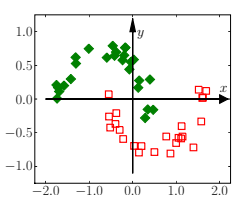
SVM nonlinear classification

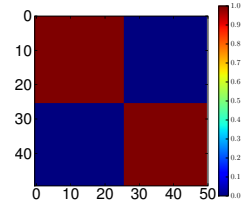
- **The solution is sparse:** only few examples $\mathbf{x}_i$ with $\alpha_i \neq 0$ are important
- **Support vectors:** define the margin and are misclassified examples

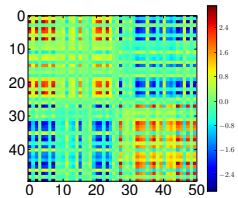


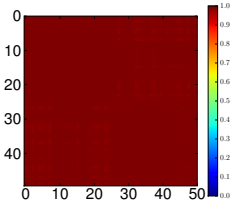
SVM nonlinear classification

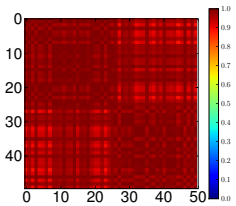
- Large $C \rightarrow$ overfitting
- Small $C \rightarrow$ oversmooth
- Choosing σ for the RBF kernel is critical, as it indicates the degree of shared information among training samples

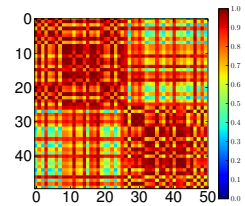

two moons

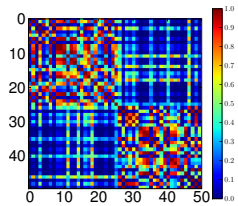

label "kernel"

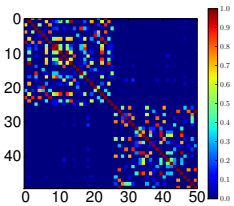

linear

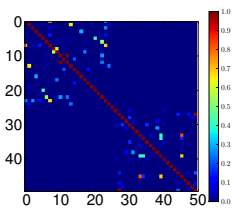

Gauss: $\gamma = 0.001$

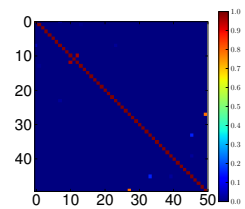

$\gamma = 0.01$

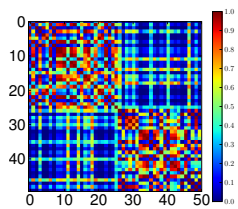

$\gamma = 0.1$

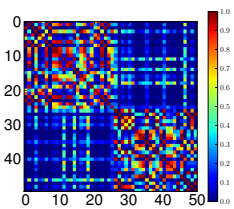

$\gamma = 1$


$\gamma = 10$


$\gamma = 100$


$\gamma = 1000$


$\gamma = 0.6$
rule of thumb


$\gamma = 1.6$
5-fold CV

SVM nonlinear classification

Hands on. From Theory to Computer

- We only need to solve this problem:
  1. Compute the kernel matrix with all labeled samples, $\mathbf{K}$.
  2. Solve the problem:

$$\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2} \boldsymbol{\alpha}^\top \mathbf{YKY} \boldsymbol{\alpha} - \mathbf{1}^\top \boldsymbol{\alpha} \right\}$$

subject to:

$$0 \leq \boldsymbol{\alpha} \leq C$$

- Matlab solves it with `quadprog.m`:

```
alpha = quadprog(K,Ones,[],[],Y,0,0,C);
```

- Visit `http://www.kernel-machines.org`
- Many SVM implementations: Pegasus, SVM^{light}, MySVM, SVMTorch,
- Machine learning tools: Weka, Spider, Shogun
- A fast C++ implementation is `libSVM` by Chih-Jen Li
- A tuned (and even faster implementation):

`http://www.uv.es/jordi`

Motivation

- Clustering is a key problem in nature
- Learn to group examples in meaningful families/clusters/groups without training labels
- Clustering is difficult and ill-posed

Motivation

- Clustering is a key problem in nature
- Learn to group examples in meaningful families/clusters/groups without training labels
- Clustering is difficult and ill-posed

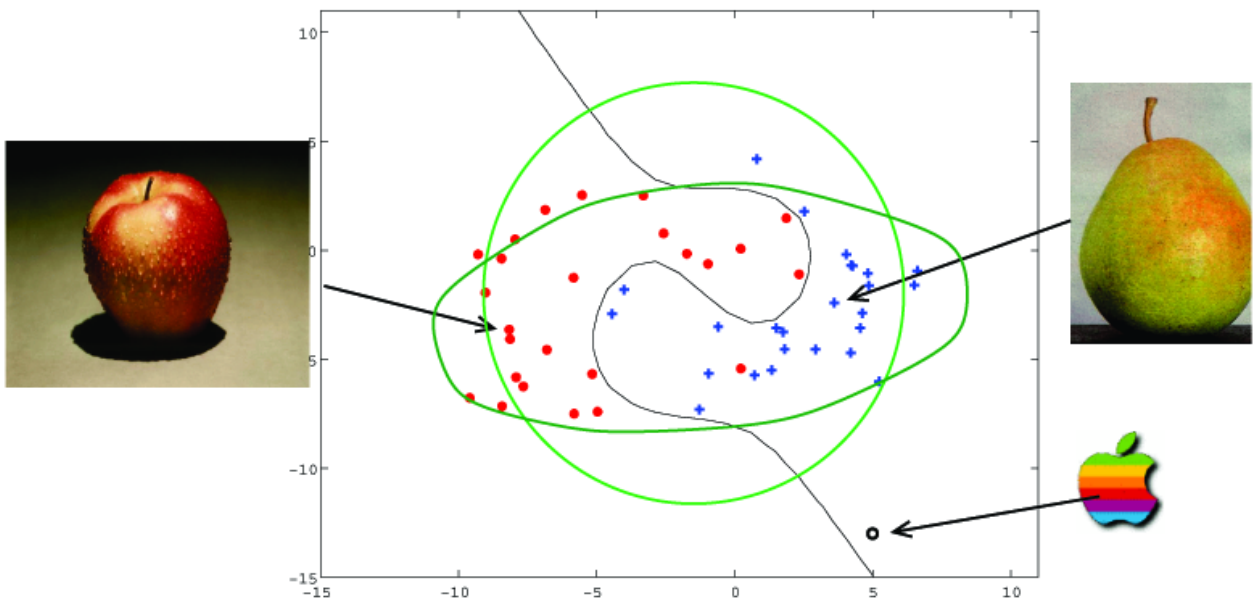
Outline

- What do we need for clustering? Just measure distances!
- Kernel methods can do the job: kernelization modes

Introduction to one-class classification

One-class classification

- A.k.a. target detection, anomaly detection, outlier detection, etc.
- Sometimes you're only interested in detecting one class and rejecting the others
- Sometimes you have labels for just one class
- Somehow related to density estimation!



Introduction to one-class classification

Main ideas

1. Boundary between the two classes has to be estimated from data of only one class
2. Task: to define a boundary around the target class (to accept as much of the target objects as possible, to minimize the chance of accepting outlier objects).

Introduction to one-class classification

Main ideas

1. Boundary between the two classes has to be estimated from data of only one class
2. Task: to define a boundary around the target class (to accept as much of the target objects as possible, to minimize the chance of accepting outlier objects).

Examples

- Speech: identification of intruders
- Computer vision: image retrieval, steganalysis, intrusion detection, etc.
- Remote sensing:
  - detect changes in two images,
  - identify one class: urban, cloud, citrus trees
  - compare new examples to a library of well-characterized objects
- Bioinformatics: detect anomalous proteins in a sample
- Econometrics: find anomalous exchange rates
- Image processing: find salient, rare image features
- Online, incremental learning

Kernelizing clustering algorithms

Family 1: Kernelization of the metric

Methods based on kernelization of the metric look for centroids in input space and the distances between patterns and centroids is computed by means of kernels:

$$\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 = K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)$$

Kernelizing clustering algorithms

Family 1: Kernelization of the metric

Methods based on kernelization of the metric look for centroids in input space and the distances between patterns and centroids is computed by means of kernels:

$$\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 = K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)$$

Family 2: Description via support vectors

- The description via support vectors makes use of One Class SVM to find a minimum enclosing sphere in feature space able to enclose almost all data in feature space excluding outliers.
- The computed hypersphere corresponds to nonlinear surfaces in input space enclosing groups of patterns.

Kernelizing clustering algorithms

Family 1: Kernelization of the metric

Methods based on kernelization of the metric look for centroids in input space and the distances between patterns and centroids is computed by means of kernels:

$$\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 = K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)$$

Family 2: Description via support vectors

- The description via support vectors makes use of One Class SVM to find a minimum enclosing sphere in feature space able to enclose almost all data in feature space excluding outliers.
- The computed hypersphere corresponds to nonlinear surfaces in input space enclosing groups of patterns.

Family 3: Clustering in feature space

- Clustering in feature space is made by mapping each pattern using ϕ and then computing centroids $\mathbf{c}_k$ in feature space.
- It is possible to compute $\|\phi(\mathbf{x}_i) - \boldsymbol{\mu}_k\|^2$ by means of the kernel trick

Kernel novelty detector (KND)

- Put a ball around the center of mass $\phi_\mu \in \mathcal{H}$ of enough radius to contain all the data:

$$\|\phi(\mathbf{x}_*) - \phi_\mu\| \geq \max_{1 \leq i \leq n} \{\|\phi(\mathbf{x}_i) - \phi_\mu\|\}$$

Kernel novelty detector (KND)

- Put a ball around the center of mass $\phi_\mu \in \mathcal{H}$ of enough radius to contain all the data:

$$\|\phi(\mathbf{x}_*) - \phi_\mu\| \geq \max_{1 \leq i \leq n} \{\|\phi(\mathbf{x}_i) - \phi_\mu\|\}$$

- This condition can be tested with kernels because we can compute distances:

$$d_{\mathcal{H}}(\mathbf{x}_i, \mathbf{x}_j) = \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|_{\mathcal{H}} = \sqrt{K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)}$$

Kernel novelty detector (KND)

- Put a ball around the center of mass $\phi_\mu \in \mathcal{H}$ of enough radius to contain all the data:

$$\|\phi(\mathbf{x}_*) - \phi_\mu\| \geq \max_{1 \leq i \leq n} \{\|\phi(\mathbf{x}_i) - \phi_\mu\|\}$$

- This condition can be tested with kernels because we can compute distances:

$$d_{\mathcal{H}}(\mathbf{x}_i, \mathbf{x}_j) = \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|_{\mathcal{H}} = \sqrt{K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)}$$

- ... and also remember that the mean of the data is $\phi_\mu = \frac{1}{n}\sum_{i=1}^{n} \phi(\mathbf{x}_i)$, then

$$\tilde{K}(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) - \frac{2}{n}\sum_i K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{n^2}\sum_{i,j} K(\mathbf{x}_i, \mathbf{x}_j)$$

Kernel novelty detector (KND)

- Put a ball around the center of mass $\phi_\mu \in \mathcal{H}$ of enough radius to contain all the data:
$$\|\phi(\mathbf{x}_*) - \phi_\mu\| \geq \max_{1\leq i\leq n} \{\|\phi(\mathbf{x}_i) - \phi_\mu\|\}$$
- This condition can be tested with kernels because we can compute distances:
$$d_{\mathcal{H}}(\mathbf{x}_i, \mathbf{x}_j) = \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|_{\mathcal{H}} = \sqrt{K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)}$$
- ... and also remember that the mean of the data is $\phi_\mu = \frac{1}{n}\sum_{i=1}^{n} \phi(\mathbf{x}_i)$, then
$$\tilde{K}(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) - \frac{2}{n}\sum_i K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{n^2}\sum_{i,j} K(\mathbf{x}_i, \mathbf{x}_j)$$
- Prove that the KND reduces to test this condition:
$$K(\mathbf{x}_*, \mathbf{x}_*) + \frac{1}{n^2}\sum_{i,j} K(\mathbf{x}_i, \mathbf{x}_j) - \frac{2}{n}\sum_i K(\mathbf{x}_*, \mathbf{x}_j)$$
$$\geq$$
$$\max_{1\leq i\leq n} \left\{ K(\mathbf{x}_i, \mathbf{x}_i) + \frac{1}{n^2}\sum_{i,j} K(\mathbf{x}_i, \mathbf{x}_j) - \frac{2}{n}\sum_i K(\mathbf{x}_i, \mathbf{x}_j) \right\}$$

Enhanced Kernel novelty detectors (KND)

- Put a ball around the center of mass $\phi_\mu \in \mathcal{H}$ of enough radius to contain all the data:

$$\|\phi(\mathbf{x}_*) - \phi_\mu\| \geq \max_{1 \leq i \leq n} \{\|\phi(\mathbf{x}_i) - \phi_\mu\|\}$$

Enhanced Kernel novelty detectors (KND)

- Put a ball around the center of mass $\phi_\mu \in \mathcal{H}$ of enough radius to contain all the data:

$$\|\phi(\mathbf{x}_*) - \phi_\mu\| \geq \max_{1 \leq i \leq n} \{\|\phi(\mathbf{x}_i) - \phi_\mu\|\}$$

- Give more weight to the recent samples? Include a forgetting factor...

$$\|\phi(\mathbf{x}_*) - \phi_\mu\| \geq \max_{1 \leq i \leq n} \{\|\phi(\mathbf{x}_i) - \lambda^{-n}\phi_\mu\|\}$$

Enhanced Kernel novelty detectors (KND)

- Put a ball around the center of mass $\phi_\mu \in \mathcal{H}$ of enough radius to contain all the data:
$$\|\phi(\mathbf{x}_*) - \phi_\mu\| \geq \max_{1 \leq i \leq n} \{\|\phi(\mathbf{x}_i) - \phi_\mu\|\}$$
- Give more weight to the recent samples? Include a forgetting factor...
$$\|\phi(\mathbf{x}_*) - \phi_\mu\| \geq \max_{1 \leq i \leq n} \{\|\phi(\mathbf{x}_i) - \lambda^{-n}\phi_\mu\|\}$$
- Incorporate the variance of the ball somehow?
$$\phi_\sigma = \frac{1}{n}\sum_{i=1}^{n}(\phi(\mathbf{x}_i) - \phi_\mu)^2$$

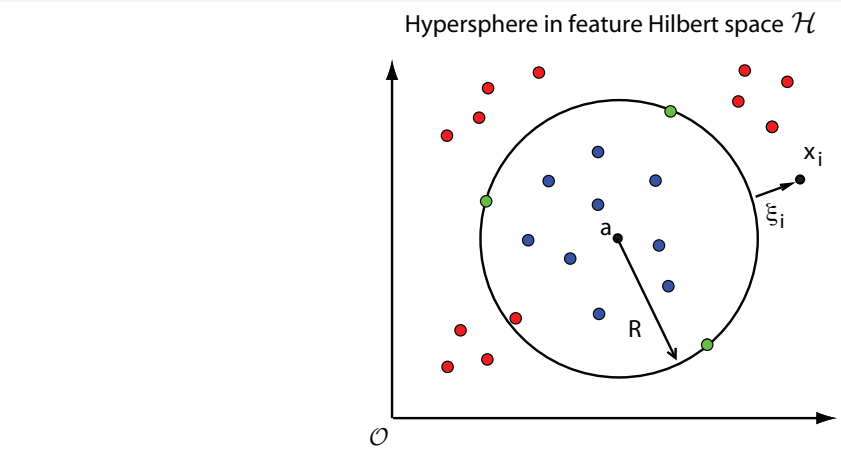
Enhanced Kernel novelty detectors (KND)

- Put a ball around the center of mass $\phi_\mu \in \mathcal{H}$ of enough radius to contain all the data:
$$\|\phi(\mathbf{x}_*) - \phi_\mu\| \geq \max_{1 \leq i \leq n} \{\|\phi(\mathbf{x}_i) - \phi_\mu\|\}$$
- Give more weight to the recent samples? Include a forgetting factor...
$$\|\phi(\mathbf{x}_*) - \phi_\mu\| \geq \max_{1 \leq i \leq n} \{\|\phi(\mathbf{x}_i) - \lambda^{-n}\phi_\mu\|\}$$
- Incorporate the variance of the ball somehow?
$$\phi_\sigma = \frac{1}{n}\sum_{i=1}^{n}(\phi(\mathbf{x}_i) - \phi_\mu)^2$$
- Update the rule? For any incoming $\mathbf{x}_i$, modify ϕ_μ and ϕ_σ

Formulation

One-class SVM

- Also known as 'support vector domain description' (SVDD) [Tax99,Schölkopf99]
- Now we only have a dataset $\{\mathbf{x}_i\}_{i=1}^n$ belonging to a given *class of interest*
- **Goal:** "find a minimum volume *hypersphere* in a high dimensional feature space $\mathcal{H}$, with radius $R > 0$ and center $\mathbf{a} \in \mathcal{H}$, which contains most of these data objects"
- This implies:
  - Minimize the radius
  - Force all samples be inside the ball
  - Allow some errors, o.w. outliers are never rejected



Formulation

One-class SVM, formulation

- Minimize the radius: $\min\{R^2\}$

Formulation

One-class SVM, formulation

- Minimize the radius: $\min\{R^2\}$
- Force all samples be inside the ball: $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \leq R^2$

Formulation

One-class SVM, formulation

- Minimize the radius: $\min\{R^2\}$
- Force all samples be inside the ball: $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \leq R^2$
- Allow some errors: $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \leq R^2 + \xi_i$

Formulation

One-class SVM, formulation

- Minimize the radius: $\min\{R^2\}$
- Force all samples be inside the ball: $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \leq R^2$
- Allow some errors: $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \leq R^2 + \xi_i$
- This translates into:

$$\min_{R,\mathbf{a}} \left\{ R^2 + C \sum_{i=1}^{n} \xi_i \right\}$$

s.t.

$$\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \leq R^2 + \xi_i \qquad \forall i = 1, \ldots, n \qquad (1)$$

$$\xi_i \geq 0 \qquad \forall i = 1, \ldots, n \qquad (2)$$

Formulation

One-class SVM, formulation

- Minimize the radius: $\min\{R^2\}$
- Force all samples be inside the ball: $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \leq R^2$
- Allow some errors: $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \leq R^2 + \xi_i$
- This translates into:

$$\min_{R,\mathbf{a}} \left\{ R^2 + C \sum_{i=1}^{n} \xi_i \right\}$$

s.t.

$$\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \leq R^2 + \xi_i \qquad \forall i = 1, \ldots, n \qquad (1)$$

$$\xi_i \geq 0 \qquad \forall i = 1, \ldots, n \qquad (2)$$

- Free parameters: the big problem! no xval procedures possible!
  - Kernel parameters: low sigma means no rejection, high sigma means all accepted
  - C: controls the trade-off between the volume of the hypersphere and the permitted errors
  - $\nu = 1/nC$: rejection fraction parameter (what rate of [%] samples are outside the ball)

Formulation

One-class SVM primal problem

$$\min_{R,\mathbf{a}} \left\{ R^2 + C \sum_{i=1}^{n} \xi_i \right\} \qquad s.t. \qquad \|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \leq R^2 + \xi_i \qquad \xi_i \geq 0$$

One-class SVM, solution

- Include restrictions into the primal:

$$\min_{R,\mathbf{a},\xi_i} \left\{ R^2 + C \sum_{i=1}^{n} \xi_i - \sum_i \alpha_i \left[R^2 + \xi_i - \|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \right] - \sum_i \mu_i \xi_i \right\}$$

- Derive and equal to zero:

$$\frac{\partial L}{\partial R} = 0 \rightarrow \sum_i \alpha_i = 1$$

$$\frac{\partial L}{\partial \mathbf{a}} = 0 \rightarrow \mathbf{a} = \frac{\sum_i \alpha_i \phi(\mathbf{x}_i)}{\sum_i \alpha_i}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \rightarrow C = \alpha_i + \mu_i \rightarrow 0 \leq \alpha_i \leq C$$

Formulation

One-class SVM dual problem

- Include this constraints to derive the dual problem:

$$\mathcal{L}_D = \sum_i \alpha_i \phi(\mathbf{x}_i)\phi(\mathbf{x}_i) - \sum_i \sum_j \alpha_i \alpha_j \phi(\mathbf{x}_i)\phi(\mathbf{x}_j) \qquad s.t. \qquad 0 \leq \alpha_i \leq C$$

- This is a QP problem whose solution yields a set of Lagrange multipliers α_i

Formulation

One-class SVM dual problem

- Include this constraints to derive the dual problem:

$$\mathcal{L}_D = \sum_i \alpha_i \phi(\mathbf{x}_i)\phi(\mathbf{x}_i) - \sum_i \sum_j \alpha_i \alpha_j \phi(\mathbf{x}_i)\phi(\mathbf{x}_j) \qquad s.t. \qquad 0 \leq \alpha_i \leq C$$

- This is a QP problem whose solution yields a set of Lagrange multipliers α_i
- Interpretation of the constraints:
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \leq R^2 + \xi_i$ is fulfiled then $\alpha_i = 0$
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 = R^2 + \xi_i$ then $\alpha_i > 0$
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 < R^2 \rightarrow \alpha_i = 0, \mu_i = 0$ (target samples, not SVs)
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 = R^2 \rightarrow 0 < \alpha_i < C, \mu_i = 0$ (unbounded SVs)
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 > R^2 \rightarrow \alpha_i = C, \mu_i > 0$ (bounded SVs, outliers)

Formulation

One-class SVM dual problem

- Include this constraints to derive the dual problem:

$$\mathcal{L}_D = \sum_i \alpha_i \phi(\mathbf{x}_i)\phi(\mathbf{x}_i) - \sum_i \sum_j \alpha_i \alpha_j \phi(\mathbf{x}_i)\phi(\mathbf{x}_j) \qquad s.t. \qquad 0 \leq \alpha_i \leq C$$

- This is a QP problem whose solution yields a set of Lagrange multipliers α_i
- Interpretation of the constraints:
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \leq R^2 + \xi_i$ is fulfiled then $\alpha_i = 0$
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 = R^2 + \xi_i$ then $\alpha_i > 0$
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 < R^2 \rightarrow \alpha_i = 0, \mu_i = 0$ (target samples, not SVs)
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 = R^2 \rightarrow 0 < \alpha_i < C, \mu_i = 0$ (unbounded SVs)
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 > R^2 \rightarrow \alpha_i = C, \mu_i > 0$ (bounded SVs, outliers)
- If C is properly tuned, most of the α_i are zero $\rightarrow$ Sparsity

Formulation

One-class SVM dual problem

- Include this constraints to derive the dual problem:

$$\mathcal{L}_D = \sum_i \alpha_i \phi(\mathbf{x}_i)\phi(\mathbf{x}_i) - \sum_i \sum_j \alpha_i \alpha_j \phi(\mathbf{x}_i)\phi(\mathbf{x}_j) \qquad s.t. \qquad 0 \leq \alpha_i \leq C$$

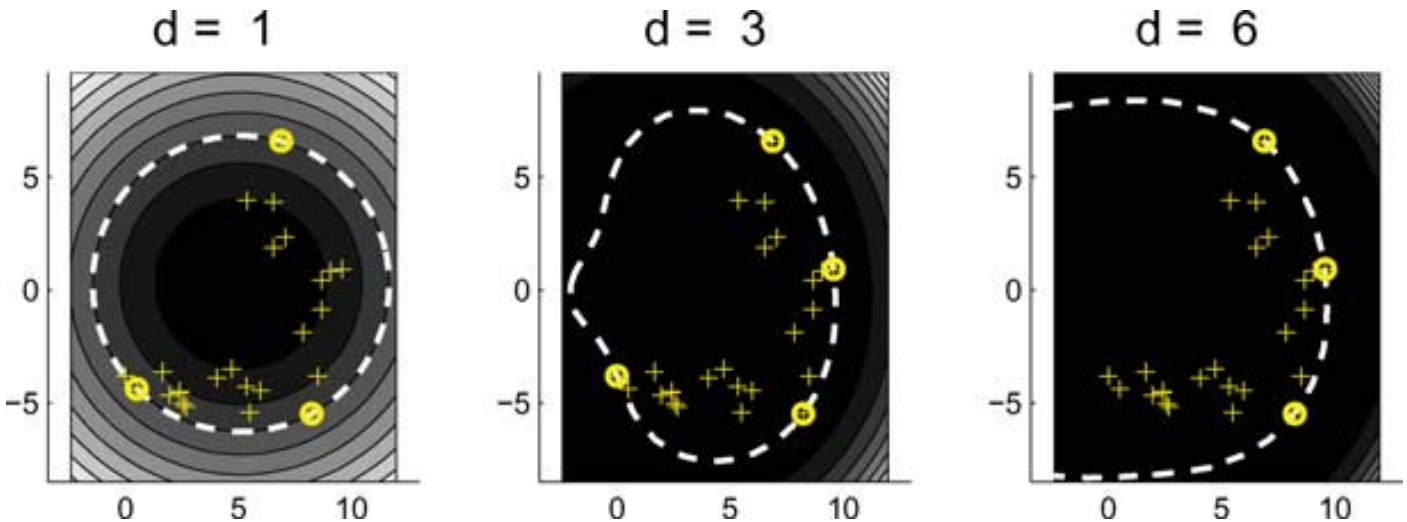
- This is a QP problem whose solution yields a set of Lagrange multipliers α_i
- Interpretation of the constraints:
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \leq R^2 + \xi_i$ is fulfiled then $\alpha_i = 0$
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 = R^2 + \xi_i$ then $\alpha_i > 0$
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 < R^2 \rightarrow \alpha_i = 0, \mu_i = 0$ (target samples, not SVs)
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 = R^2 \rightarrow 0 < \alpha_i < C, \mu_i = 0$ (unbounded SVs)
  - if $\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 > R^2 \rightarrow \alpha_i = C, \mu_i > 0$ (bounded SVs, outliers)
- If C is properly tuned, most of the α_i are zero $\rightarrow$ Sparsity

One-class SVM prediction function

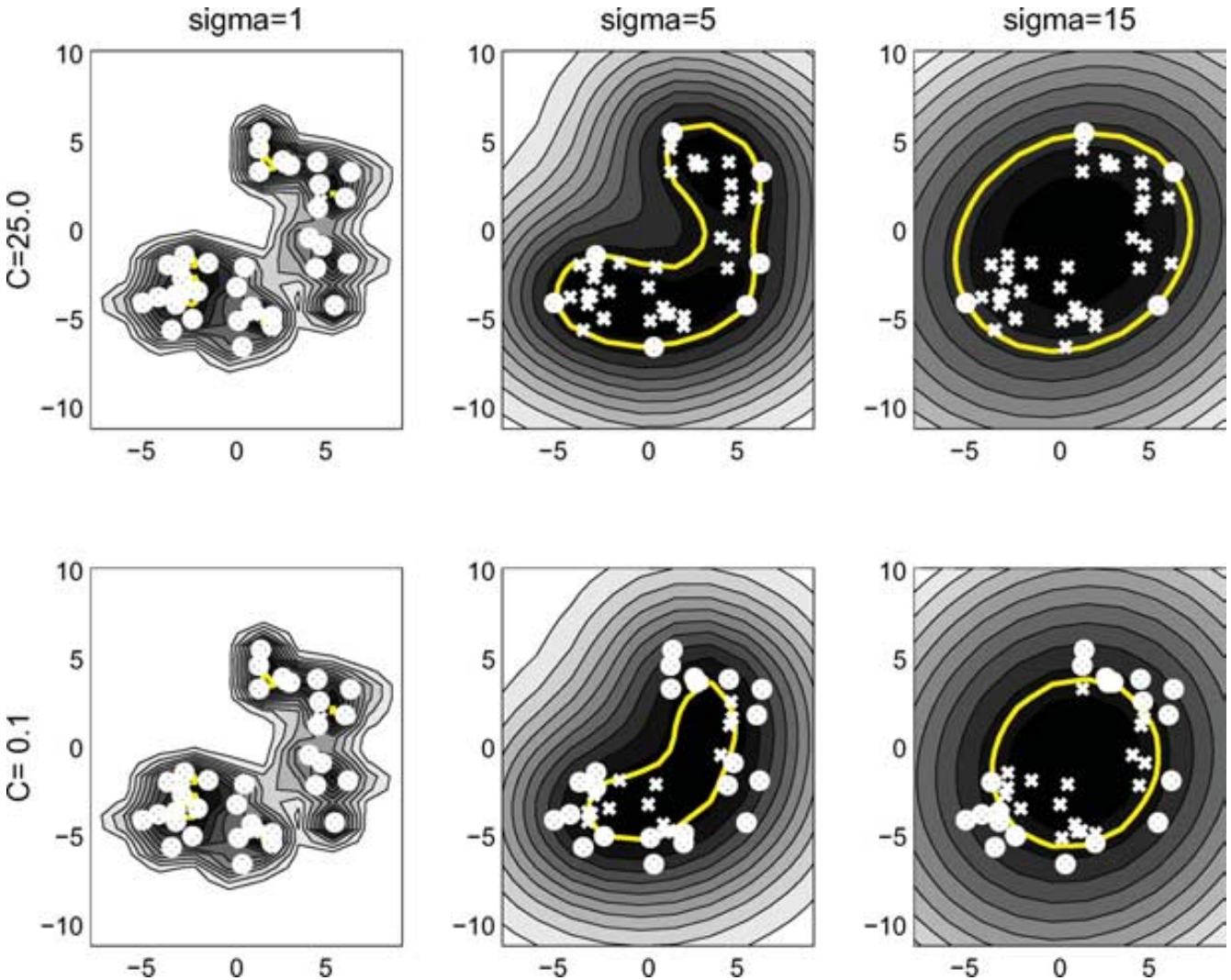
To test a new sample $\mathbf{x}_*$, we compute the distance to the center of the sphere

$$d(\mathbf{x}_*, \mathbf{a})^2 = \|\phi(\mathbf{x}_*) - \mathbf{a}\|^2 = K(\mathbf{x}_*, \mathbf{x}_*) - 2\sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}_*) + \sum_{i,j=1}^{n} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) > R^2$$

Effect of the free parameters



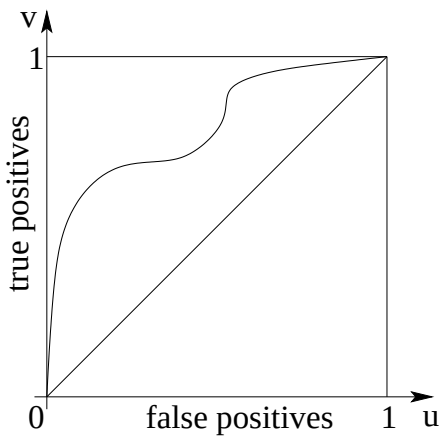
Effect of the free parameters



Effect of the free parameters

How to tune the parameters without outliers?

- Put a reasonable σ (median better than the mean of the distance between samples)
- If you have an estimation of the rate of expected outliers, tune $\nu = 1/nC$
- Typically: $\nu \in [0.1, 0.4]$ (force sparsity)
- Heuristic: maximize the rate $SVs/errors$
- Heuristic 2: update parameters for a new incoming sample
- Analyze the receiver operating curve (ROC): $f(u, v|\theta)$
  - u = proportion of false positives = $P(f(x) = 1|y = -1)$
  - v = proportion of true positives = $P(f(x) = 1|y = 1)$



Effect of the free parameters

Hands on. From Theory to Computer

- We only need to solve this problem:
  1. Compute the kernel matrix with all samples, $\mathbf{K}$.
  2. Solve the problem:

$$\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2} \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} - \mathbf{1}^\top \boldsymbol{\alpha} \right\}$$

subject to:

$$0 \leq \boldsymbol{\alpha} \leq C$$

- Matlab solves it with `quadprog.m`:

```
alpha = quadprog(K,Ones,[],[],Ones,0,0,C);
```

- Visit `http://www.kernel-machines.org`
- Many SVM implementations: Pegasus, SVM^{light}, MySVM, SVMTorch,
- Machine learning tools: Weka, Spider, Shogun
- A fast C++ implementation is `libSVM` by Chih-Jen Li
- A tuned (and even faster implementation):

```
http://www.uv.es/jordi
```

- **Tax DDTools: very useful for one-class and PDE!**

k-means algorithm

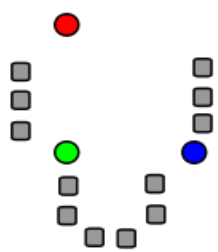
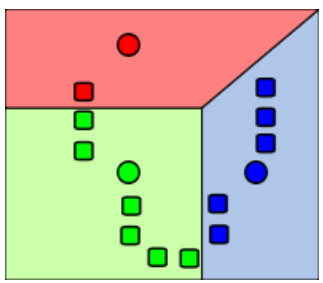
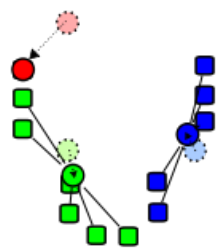
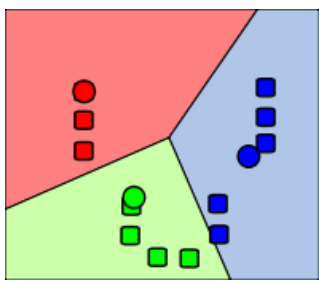
Given a set of observations $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, partition the n observations into k sets ($k \leq n$), $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares:

$$\arg\min_{\mathbf{S}} \left\{ \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2 \right\}$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i

The Lloyd's (iterative) algorithm

1. **Initialization,** $t = 0$**:** Set an initial set of k-means $\boldsymbol{\mu}_1^{(t)}, \ldots, \boldsymbol{\mu}_k^{(t)}$
2. **Assignment:** Assign each observation to the cluster with closest mean:
$$S_i^{(t)} = \{\mathbf{x}_j : \|\mathbf{x}_j - \boldsymbol{\mu}_i^{(t)}\| \leq \|\mathbf{x}_j - \boldsymbol{\mu}_{i*}^{(t)}\|, \textit{for all } i* = 1, \ldots, k\}$$
3. **Update:** Assign the centroids to the new means:
$$\boldsymbol{\mu}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j$$
4. Go to step (2) until convergence, i.e. any $\boldsymbol{\mu}_i$ changes
5. Return the feature space codebook (the centroids to use in test)

			
k initial 'means' (e.g. $k = 3$) are randomly selected from the data set	k clusters are created by associating every observation with the nearest mean	The centroid of each of the k clusters becomes the new means	Steps 2 and 3 are repeated until convergence has been reached

Algorithm

This algorithm minimizes the quantization error in feature space:

1. Project data: $\mathbf{X} \rightarrow \mathbf{\Phi}$
2. Initialize the codebook: $\mathbf{M}^{\phi} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k]^{\top}$, $\boldsymbol{\mu} \in \mathcal{H}$
3. Compute for each centroid $\boldsymbol{\mu}_k$ the set S_i
4. Update the codevectors:

$$\boldsymbol{\mu}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \phi(\mathbf{x}_j)$$

5. Go to step (3) until any $\boldsymbol{\mu}_i$ changes
6. Return the feature space codebook indices

Algorithm

This algorithm minimizes the quantization error in feature space:

1. Project data: $\mathbf{X} \rightarrow \mathbf{\Phi}$
2. Initialize the codebook: $\mathbf{M}^{\phi} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k]^{\top}$, $\boldsymbol{\mu} \in \mathcal{H}$
3. Compute for each centroid $\boldsymbol{\mu}_k$ the set S_i
4. Update the codevectors:

$$\boldsymbol{\mu}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \phi(\mathbf{x}_j)$$

5. Go to step (3) until any $\boldsymbol{\mu}_i$ changes
6. Return the feature space codebook indices

Solution

1. Apply the Representer's theorem to codevectors: $\boldsymbol{\mu}_j = \frac{1}{|S_k|} \sum_{\mathbf{x}_i \in S_k} \alpha_i \phi(\mathbf{x}_i)$
2. One can compute distances in $\mathcal{H}$: $\|\phi(\mathbf{x}_i) - \boldsymbol{\mu}_j\|^2$

Samples are mapped into a higher dimensional feature space:

input data complex and nonlinear	$\xrightarrow{\varphi(\cdot)}$	mapping in a higher dimensional feature space
		$\downarrow$
nonlinear solution in input space	$\longleftarrow$	linear learning algorithm (k-means)

Basics

- The explicit computation of the mapping $\varphi(\cdot)$ is costly and often not bearable
- The dot products $\langle \varphi(\cdot), \varphi(\cdot) \rangle$ can be replaced by $k(\cdot, \cdot)$
- The value returned by the kernel function correspond to the dot product in the (high dimensional) feature space

The kernel version

- True clusters are often not recognizable in the input space
- Perform k-means in the feature space spanned by the kernel.

$$d^2(\varphi(\mathbf{x}_i), \mathbf{m}_k) = \|\varphi(\mathbf{x}_i) - \mathbf{m}_k\|^2 \quad (1)$$

$$\text{where } \mathbf{m}_k = \frac{1}{|\pi_k|} \sum_{j \in \pi_k} \varphi(\mathbf{x}_j) \quad (2)$$

In other words:

$$d^2(\varphi(\mathbf{x}_i), \mathbf{m}_k) = k(\mathbf{x}_i, \mathbf{x}_i) + \frac{1}{|\pi_k|^2} \sum_{j,l \in \pi_k} k(\mathbf{x}_j, \mathbf{x}_l) - \frac{2}{|\pi_k|} \sum_{j \in \pi_k} k(\mathbf{x}_i, \mathbf{x}_j)$$

...but some problems arise

1. As k-means, the convergence of the kernel version is greatly influenced by the initial assignments (initial centers or initial labeling)
2. The kernel function need some parameters to be tuned. How to do it keeping the algorithm unsupervised?

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle \quad k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2)$$

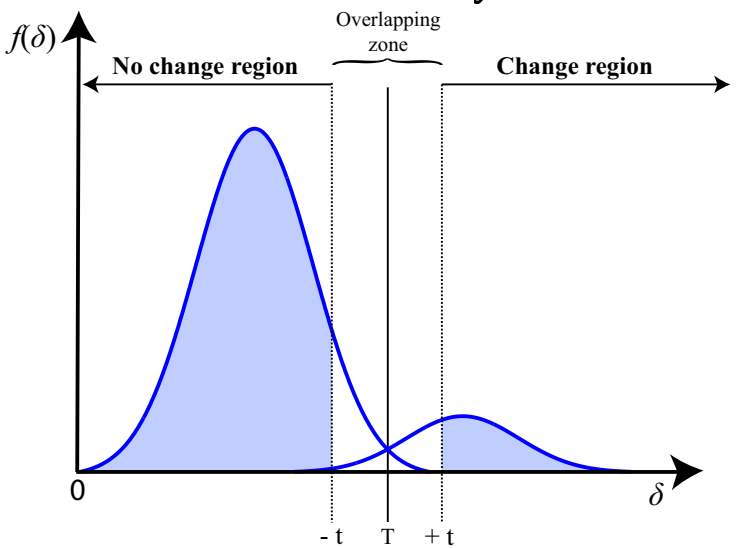
linear k-means σ need to be tuned

1. The initialization

Find some 'training' pixels

- Exploit the magnitude of the difference image in order to find some change / no change pixels
- The distribution of the magnitude of such image can be used!

Thus, *ideally*:



2. Tuning the kernel parameters

Find the correct hyperparameters:

- Map samples into compact clusters (sphericity)
- Enforce cluster separation (far centers)
- Combination of two criteria

$$\arg\min_{\sigma} \sum_{k} \frac{1}{|\pi_k|} \sum_{i \in \pi_k} d_{\sigma}^2(\varphi(\mathbf{x}_i), \mathbf{m}_k)$$

2. Tuning the kernel parameters

Find the correct hyperparameters:

- Map samples into compact clusters (sphericity)
- Enforce cluster separation (far centers)
- Combination of two criteria

$$\arg\max_{\sigma} \sum_{k \neq p} d_{\sigma}^{2}(\mathbf{m}_k, \mathbf{m}_p)$$

2. Tuning the kernel parameters

Find the correct hyperparameters:

- Map samples into compact clusters (sphericity)
- Enforce cluster separation (far centers)
- Combination of two criteria

$$\arg\min_{\sigma} \frac{\sum_k \frac{1}{|\pi_k|} \sum_{i \in \pi_k} d_\sigma^2(\varphi(\mathbf{x}_i), \mathbf{m}_k)}{\sum_{k \neq p} d_\sigma^2(\mathbf{m}_k, \mathbf{m}_p)}$$

The Zurich VHR

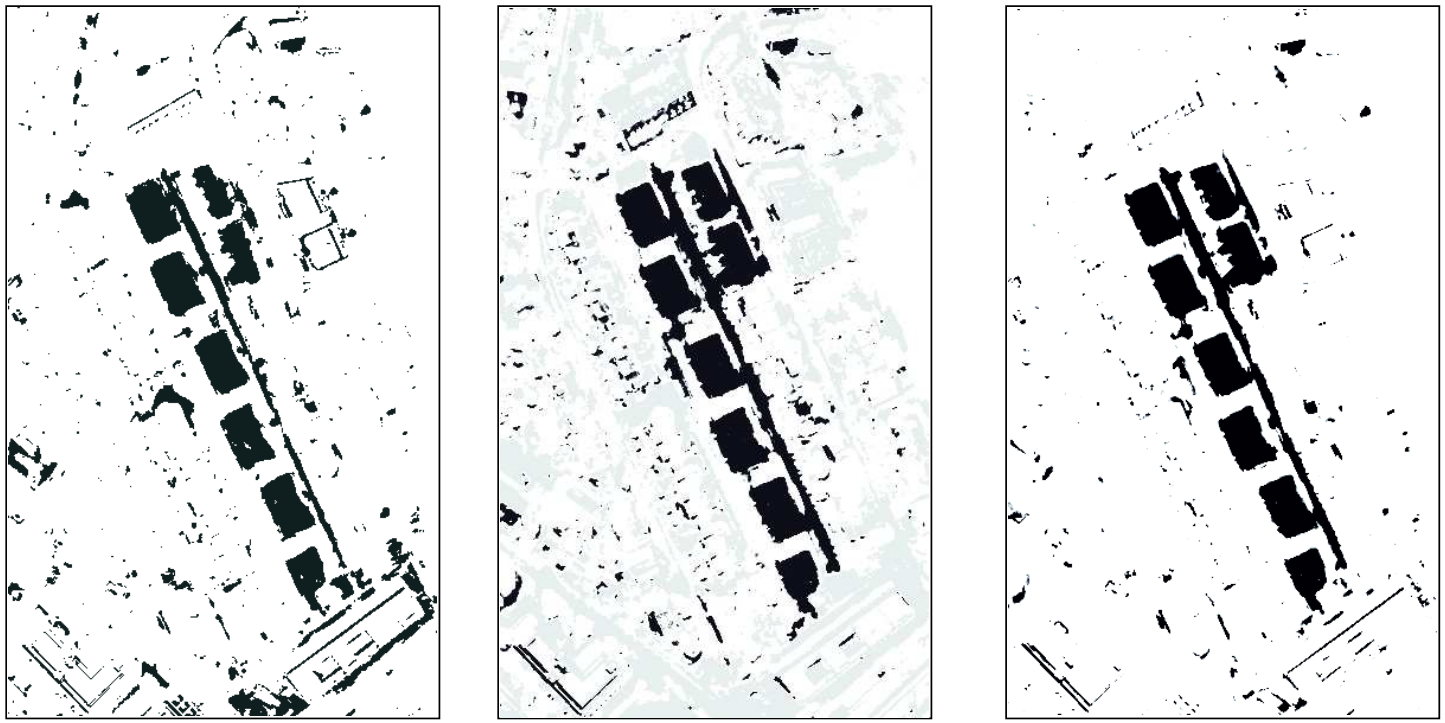
- QuickBird 0.5 m pixel size

1. Compute the magnitude $\delta = \|\mathbf{x}_{t_2} - \mathbf{x}_{t_1}\|$
2. Select pixels from both distributions, considering high noise and false change in the threshold zone
3. Find the correct parameter (here, the σ of a Gaussian RBF kernel)

ZH t_1 (2002)

ZH t_2 (2006)

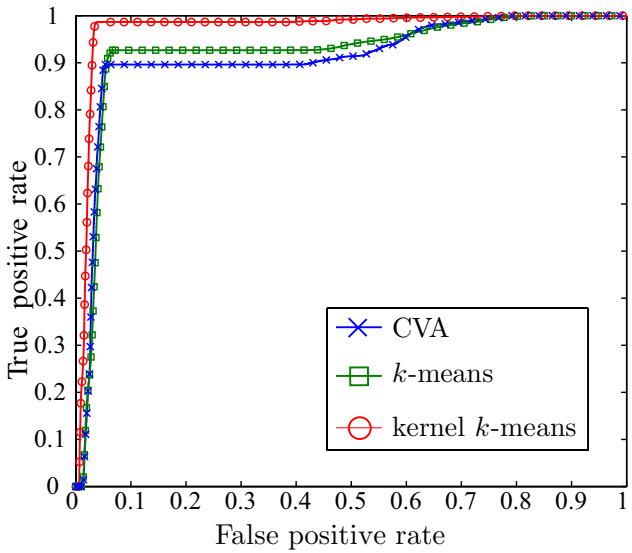
Results: linear vs. nonlinear



(a) CVA (b) k-means (c) kernel k-means

For k-means and kernel k-means these maps are a sum of 15 binary maps randomly initialized on the thresholded distributions.

ROCs



- False alarms reduced by $\sim 2\%$ for the k-means (wrt CVA) $\sim 41\%$ for the kernel k-means (wrt CVA)
- Hit rate is high for all the approaches

AUC		Cohen's Kappa	
CVA =	0.912	κ =	0.57
k-means =	0.923	κ =	0.62
kernel k-means =	0.974	κ =	0.74

Conclusions

Conclusions

- Given definition of the most useful kernel supervised and unsupervised classifiers
- Other classifiers are available: KFDA, kernel SOM, kernel fuzzy means, etc.
- Analyzed how to derive the equations
- Everything relies on the proper definition of $\mathbf{K}$

References

- J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.
- B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.
- Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, NY, 1995.
- Vladimir Vapnik. *Statistical Learning Theory*. Wiley, NY, 1998.
- Ralf Herbrich. Learning Kernel Classifiers. MIT Press, Cambridge, MA, 2002.
- J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific Pub. Co., Singapore, 2002 `http://www.esat.kuleuven.be/sista/lssvmlab/`
- Tax, David M. J. *One-class classification*, Delft University of Technology, June, Delft, (2001). `http://prlab.tudelft.nl`
- G. Camps-Valls, J. L. Rojo and M. Martinez, *Kernel Methods in Bioengineering, Signal and Image Processing*, Idea Inc., 2007.
- Thomas Hofmann, Bernhard Schlkopf, and Alexander J. Smola. *Kernel methods in machine learning.* Ann. Statist. Volume 36, Number 3 (2008), 1171-1220.
- Conferences: NIPS, ICML, ECML, COLT, ICANN, ESANN, MLSP
- Webs: `videolectures.net`, `http://www.kernel-machines.org`