Lecture 12: Kernel Methods (Part III)

Soft-margin SVM, SVR, kPCA

Machine Learning, Summer Term 2019

Michael Tangermann Frank Hutter Marius Lindauer

University of Freiburg

Lecture Overview

SVM and Non-Separable Data

Separable hyperplane may not exist in the input space $\mathcal{X}$ for practical data. Why? 

SVM and Non-Separable Data

Separable hyperplane may not exist in the input space $\mathcal{X}$ for practical data. Why?

Possible reasons:

- high noise level may cause an overlap between classes.
- training data set contains mislabelled data points (e.g. web forms)

In these cases, it may be desirable not to enforce full separability in feature space $\mathcal{H}$ but instead to allow for *small / few* violations of the margin constraints during the SVM training!

Soft-Margin SVM

Implementation of *acceptable* violations in a soft-margin SVM in three steps:

1. introduce one so-called slack variable $\xi_i \geq 0$ per training pattern.
2. utilize the slack variables to relax the constraints for the optimization problem to:
   $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \quad \text{for all} \quad i = 1, \ldots, m.$
3. Limit the sum of the slacks by an upper bound on the training error in the objective function by minimizing:

$$\tau(\mathbf{w}, \boldsymbol{\xi}) \;=\; \frac{1}{2}\|\mathbf{w}\|^2 \;+\; C \sum_{i=1}^{m} \xi_i$$

What is the role of C? 

Soft-Margin SVM

Implementation of *acceptable* violations in a soft-margin SVM in three steps:

1. introduce one so-called slack variable $\xi_i \geq 0$ per training pattern.
2. utilize the slack variables to relax the constraints for the optimization problem to:
   $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \quad$ for all $\quad i = 1, \ldots, m.$
3. Limit the sum of the slacks by an upper bound on the training error in the objective function by minimizing:

$$\tau(\mathbf{w}, \boldsymbol{\xi}) \;=\; \frac{1}{2}\|\mathbf{w}\|^2 \;+\; C \sum_{i=1}^{m} \xi_i$$

What is the role of C? The constant C determines a trade-off between enlarging the margin and minimizing the training error. [Blackboard]

Quadratic Program for Soft-Margin SVM

Incorporating the kernel trick and reformulating using Lagrange multipliers, the quadratic program for soft-margin SVMs looks very similar to that of the hard-margin SVM:

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}} \; W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

subject to $0 \leq \alpha_i \leq C$ for all $i = 1, \ldots, m$ and $\sum_{i=1}^{m} \alpha_i y_i = 0$

Comments:

- Constant C is a regularization hyperparameter: optimize it for each data set.
- C limits the influence of a single α_i, thus not allowing a single SV to "push" upon the decision hyperplane too strongly.

Quadratic Program for Soft-Margin SVM

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}} \; W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

subject to $0 \leq \alpha_i \leq C$ for all $i = 1, \ldots, m$ and $\sum_{i=1}^{m} \alpha_i y_i = 0$

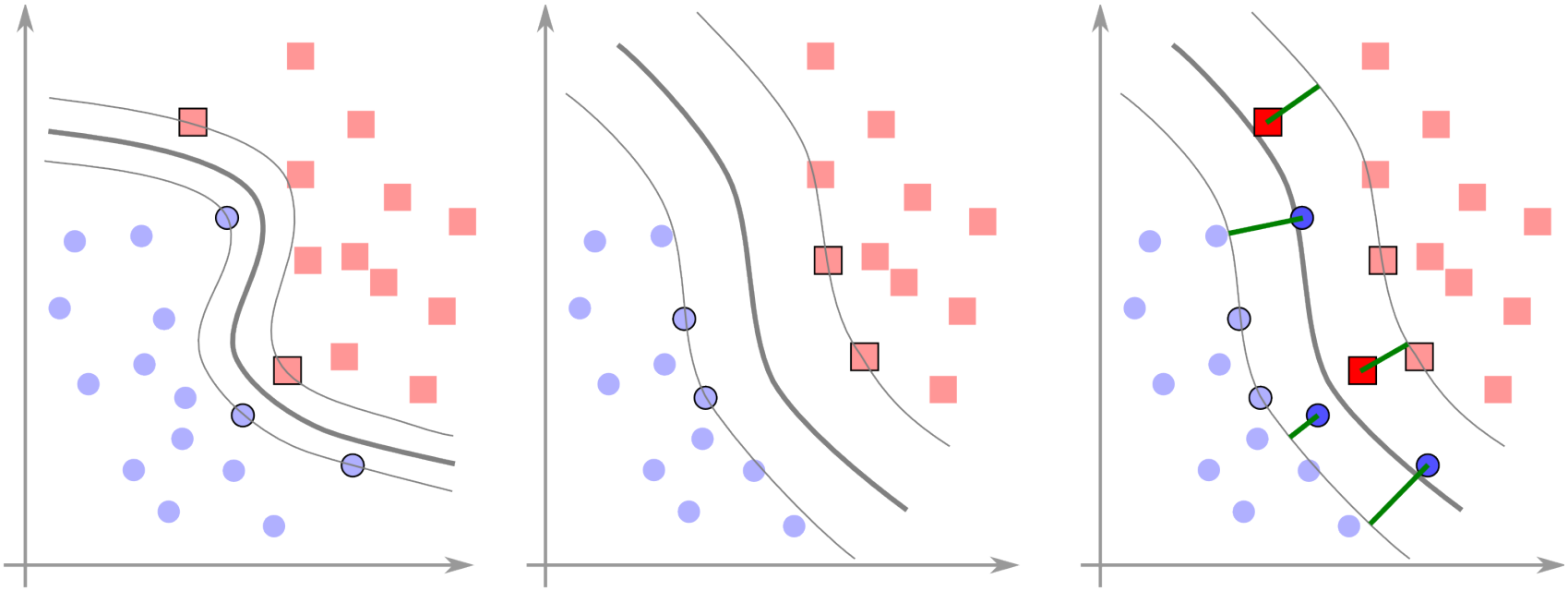
Comments (continued):

- Scalar b can be computed exploiting that for all SVs x_i with $\alpha_i < C$ the corresponding slack variable ξ_i is zero.
- Intuition: scalar b shifts the decision hyperplane such, that SVs with zero slack lie on ± 1 lines.
- A parametrization via ν (with $0 < \nu \leq 1$) is an alternative to using C and easier to handle.
- ν lower-bounds (relative to the number of training data points) the fraction of training patterns which can become SVs and thus can have non-zero slacks and it upper-bounds the fraction of margin errors (see section 7.5 of "Learning with Kernels").

Soft-Margin vs. Hard-Margin SVM for Separable Data

Even, if a (complicated) separable hyperplane exists, it may be a good idea to try soft-margin SVM in order to avoid overfitting:

- get a solution with a function class of lower capacity
- enlarge the margin ($\rightarrow$ smoother hyperplane)
- reduce R[f]



Effect of ν-Regularization on SVM for Classification

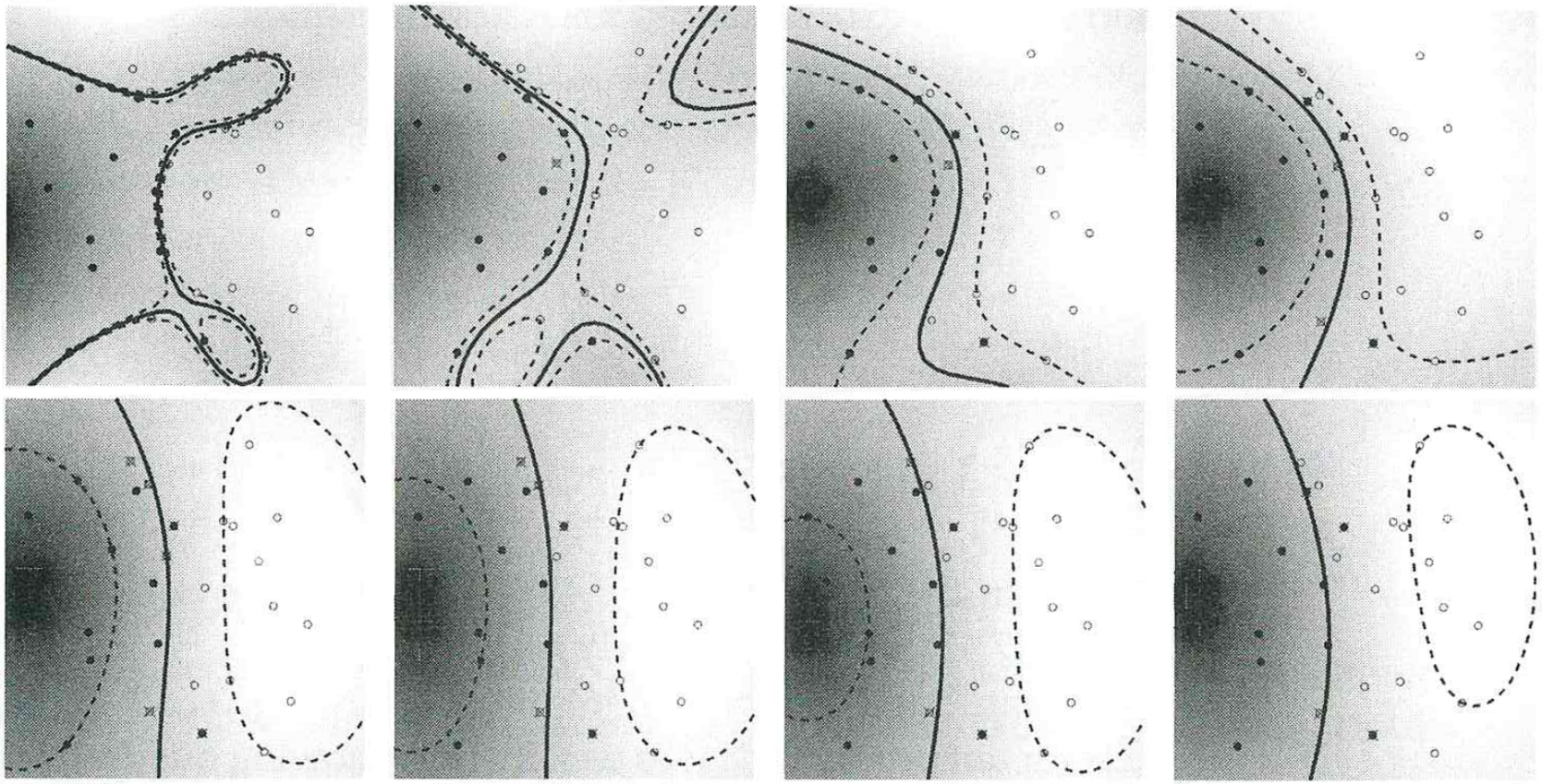


Figure 7.9 Toy problem (task: separate circles from disks) solved using ν-SV classification, with parameter values ranging from $\nu = 0.1$ (top left) to $\nu = 0.8$ (bottom right). The larger we make ν, the more points are allowed to lie inside the margin (depicted by dotted lines). Results are shown for a Gaussian kernel, $k(x, x') = \exp(-\|x - x'\|^2)$.

Kernel Parameters Can Also Regularize!

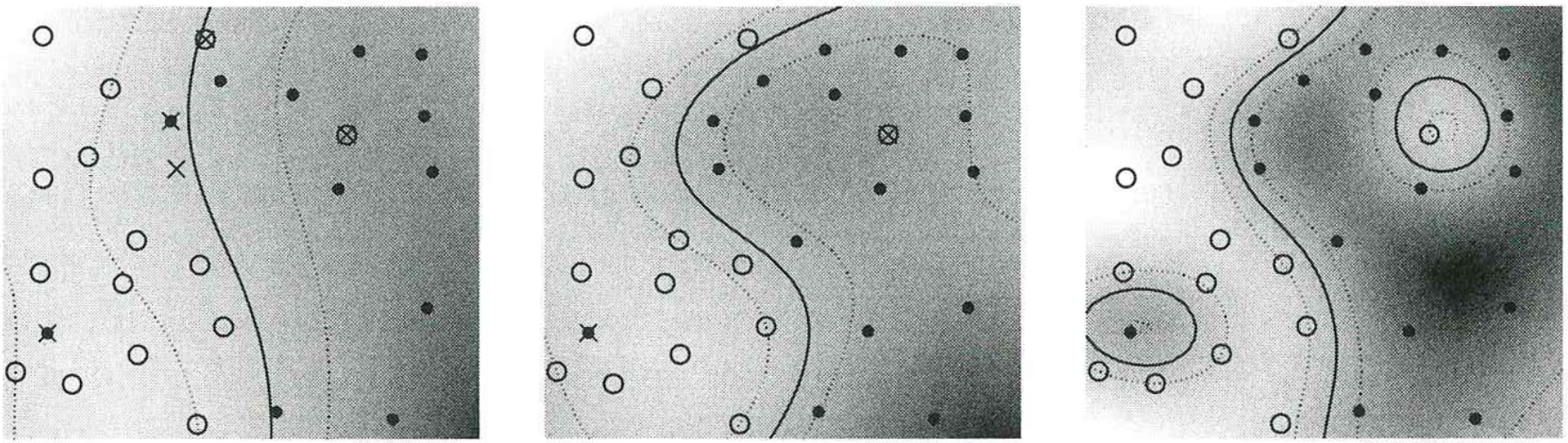
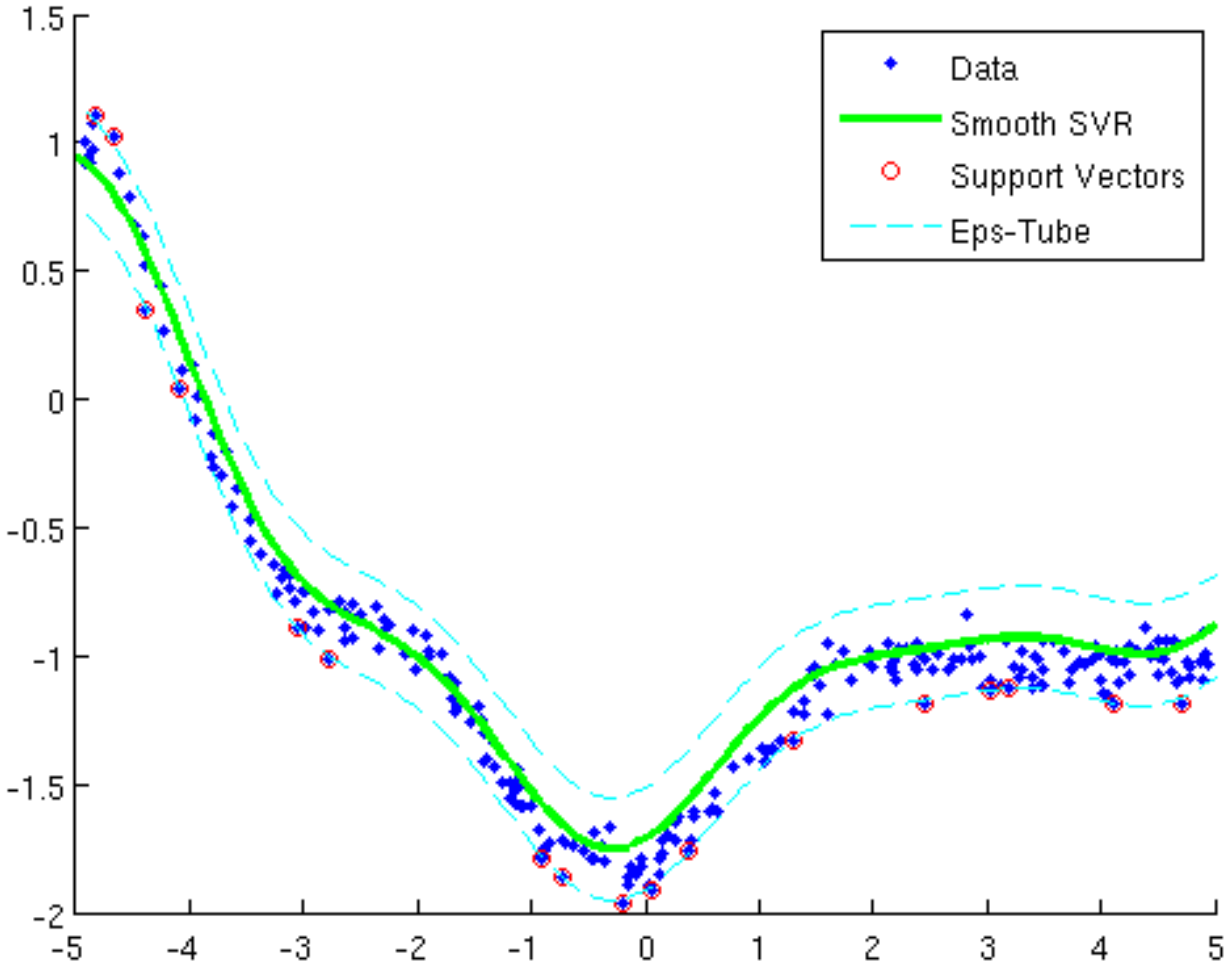


Figure 7.10 2D toy example of a binary classification problem solved using a soft margin SVC. In all cases, a Gaussian kernel (7.27) is used. From left to right, we decrease the kernel width. Note that for a large width, the decision boundary is almost linear, and the data set cannot be separated without error (see text). Solid lines represent decision boundaries; dotted lines depict the edge of the margin (where (7.34) becomes an equality with $\xi_i = 0$).

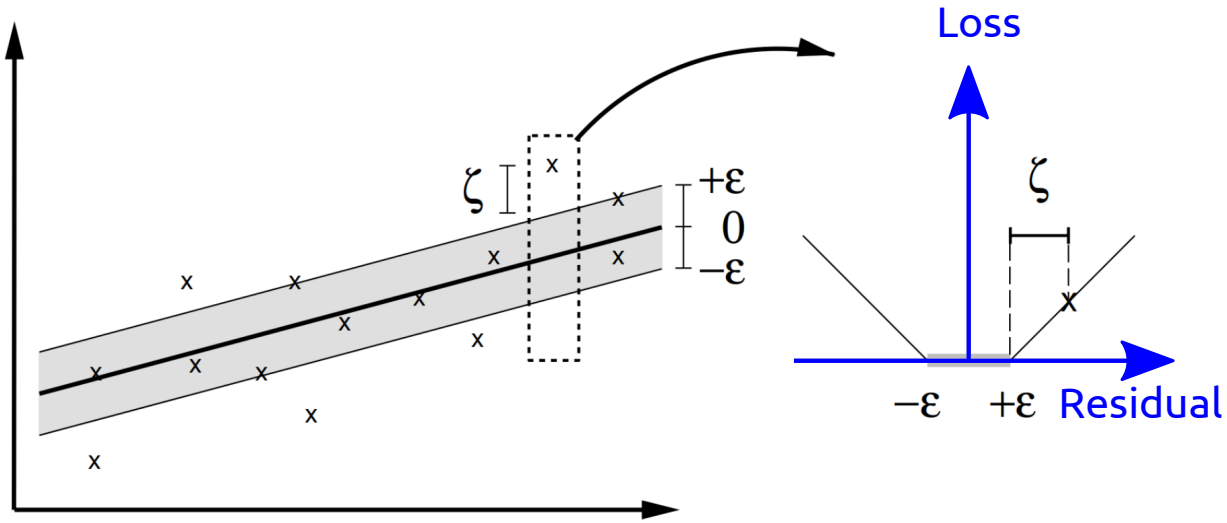
Lecture Overview

SV Method for Regression Problems

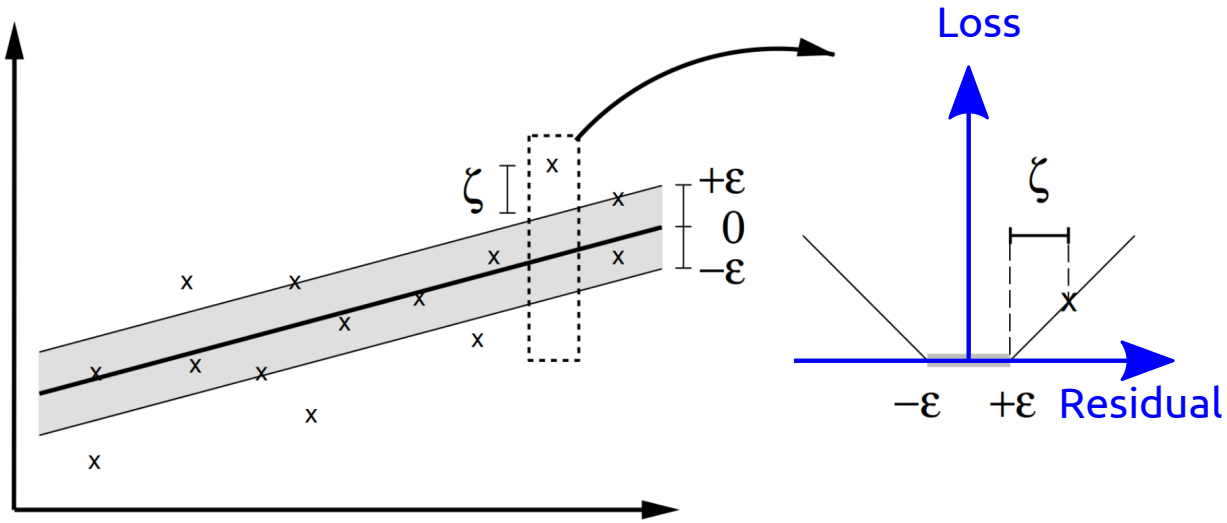
Moving from class labels $y_i \in \{\pm 1\}$ to continuous labels, a support vector regression model can estimate continuous non-linear functions.



SVM → SVR: **Margin → Tube**



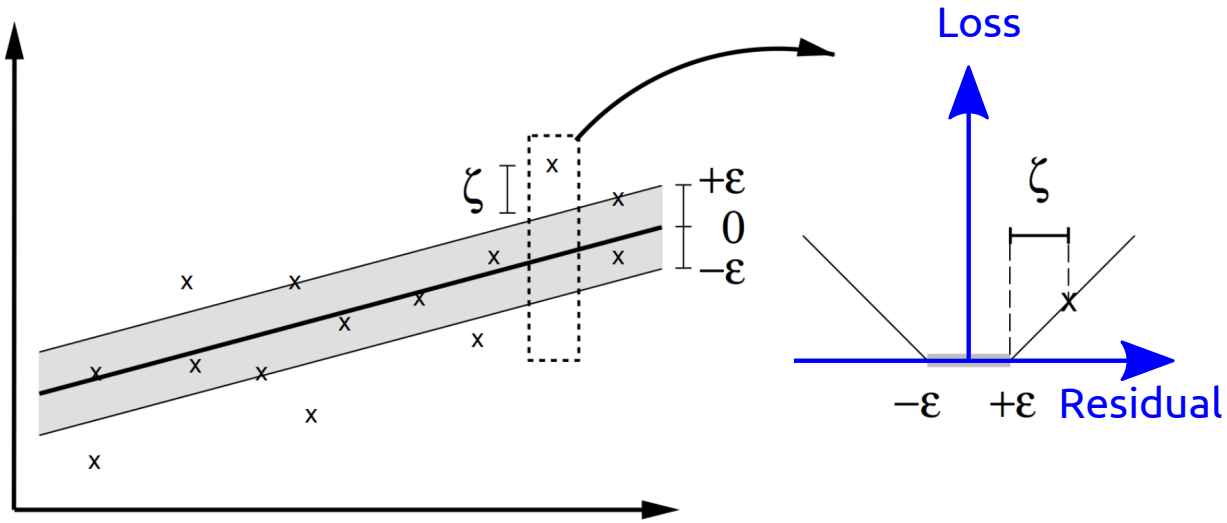
SVM → SVR: **Margin → Tube**



Key ingredients for support vector regression (SVR):

- Kernel trick (of course...)
- Use the ϵ-insensitive loss function to obtain a smooth regression. (regularizing similar to a large margin).

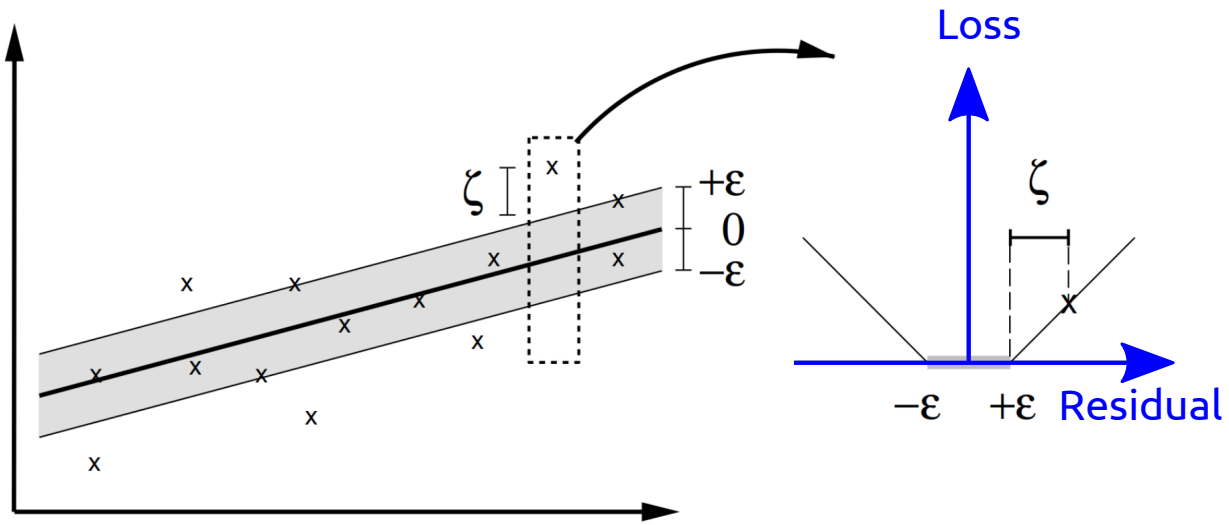
SVM → SVR: **Margin → Tube**



Key ingredients for support vector regression (SVR):

- Kernel trick (of course...)
- Use the ϵ-insensitive loss function to obtain a smooth regression. (regularizing similar to a large margin).
- Introduce (two types) of slack variables ξ_i ("xi"), which allow for violations of the ϵ tube. Limit their influence by regularization with a constant C.
- Choose $C, \epsilon \geq 0$ as hyperparameters e.g. via cross-validation.

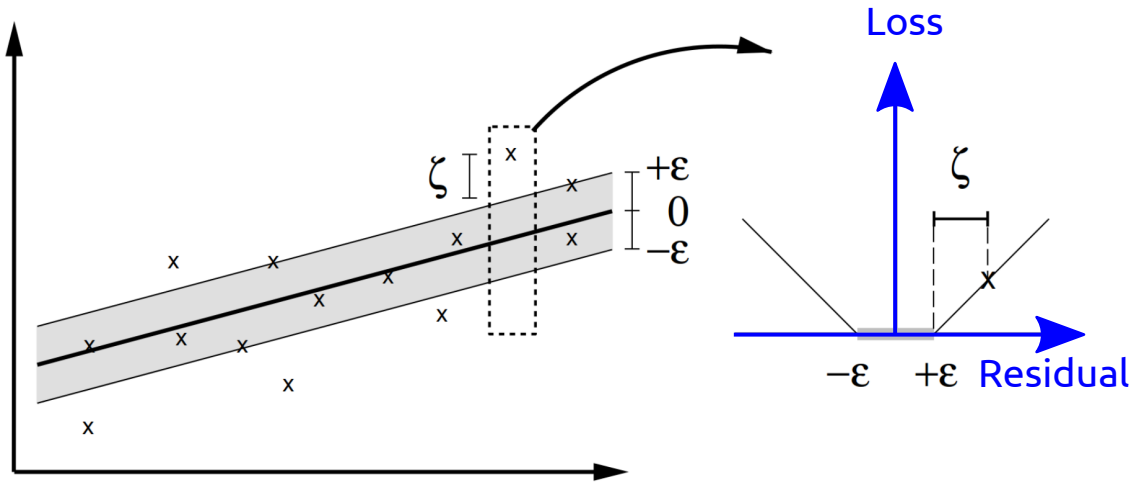
SVM → SVR: **Margin → Tube**



Key ingredients for support vector regression (SVR):

- Kernel trick (of course...)
- Use the ϵ-insensitive loss function to obtain a smooth regression. (regularizing similar to a large margin).
- Introduce (two types) of slack variables ξ_i ("xi"), which allow for violations of the ϵ tube. Limit their influence by regularization with a constant C.
- Choose $C, \epsilon \geq 0$ as hyperparameters e.g. via cross-validation.
- Formulation via Lagrange multipliers, solve quadratic problem.

Support Vector Regression (SVR)



Further reading:

- http://www.svms.org/regression/SmSc98.pdf
- Learning with Kernels, Sec. 9

Effect Of Epsilon Upon Smoothness of Regression Function

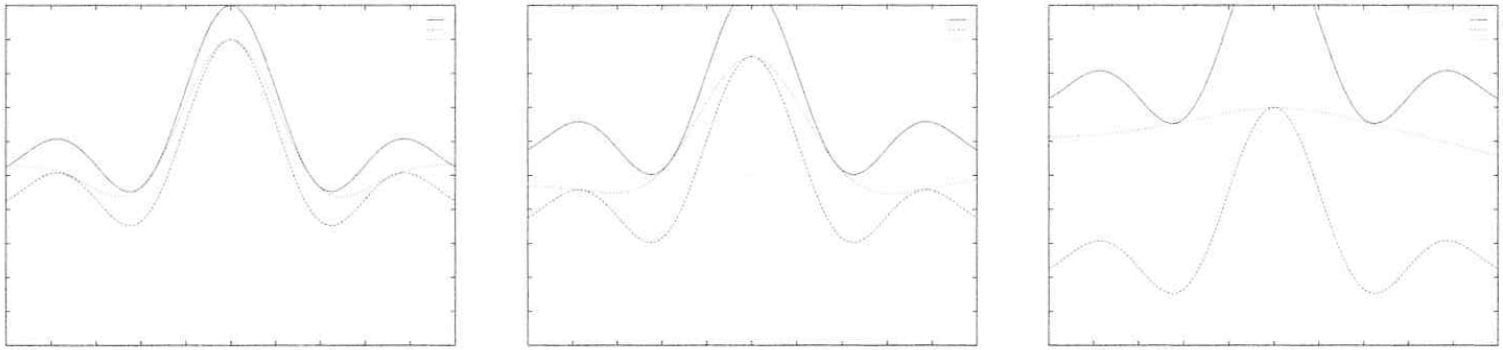


Figure 9.3 From top to bottom: approximation of the function $\operatorname{sinc} x$ with precisions $\varepsilon = 0.1, 0.2$, and 0.5. The solid top and dashed bottom lines indicate the size of the ε-tube, here drawn around the target function $\operatorname{sinc} x$. The dotted line between them is the regression function.

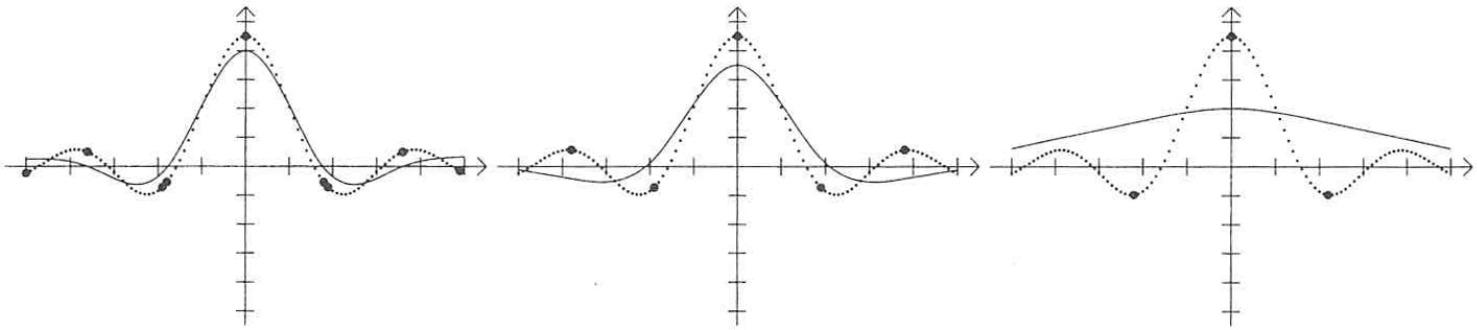


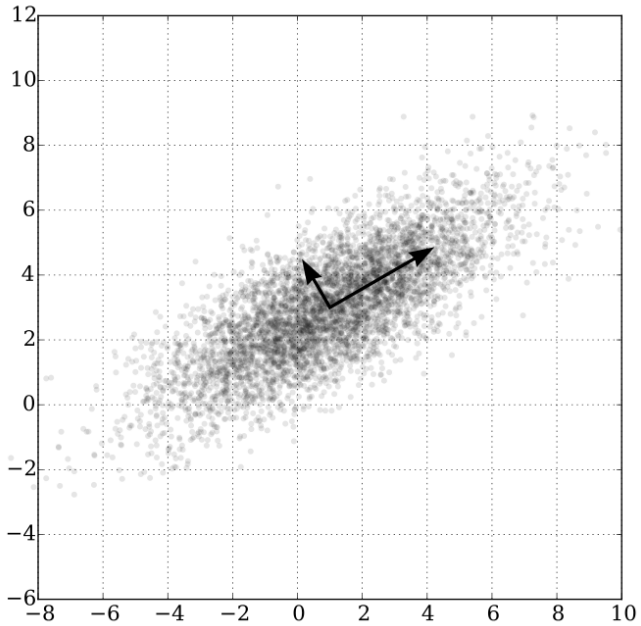
Figure 9.4 Left to right: regression (solid line), data points (small dots) and SVs (big dots) for an approximation of $\operatorname{sinc} x$ (dotted line) with $\varepsilon = 0.1, 0.2$, and 0.5. Note the decrease in the number of SVs.

Lecture Overview

Reminder Linear Principal Component Analysis

Linear principal component analysis (PCA)...

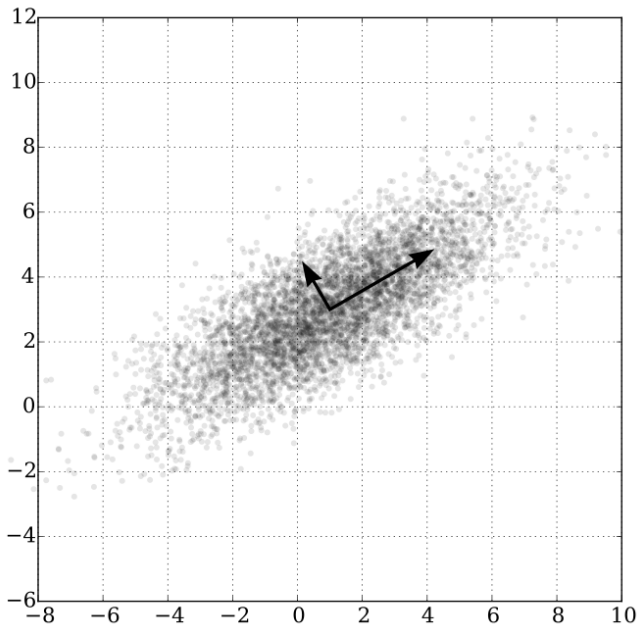
- ... obtains directions of largest variance.
- Directions are obtained as solutions (eigenvectors) of an eigenvalue problem.
- Corresponding eigenvalues deliver a sorting of the eigenvectors.



Reminder Linear Principal Component Analysis

Linear principal component analysis (PCA)...

- ... obtains directions of largest variance.
- Directions are obtained as solutions (eigenvectors) of an eigenvalue problem.
- Corresponding eigenvalues deliver a sorting of the eigenvectors.

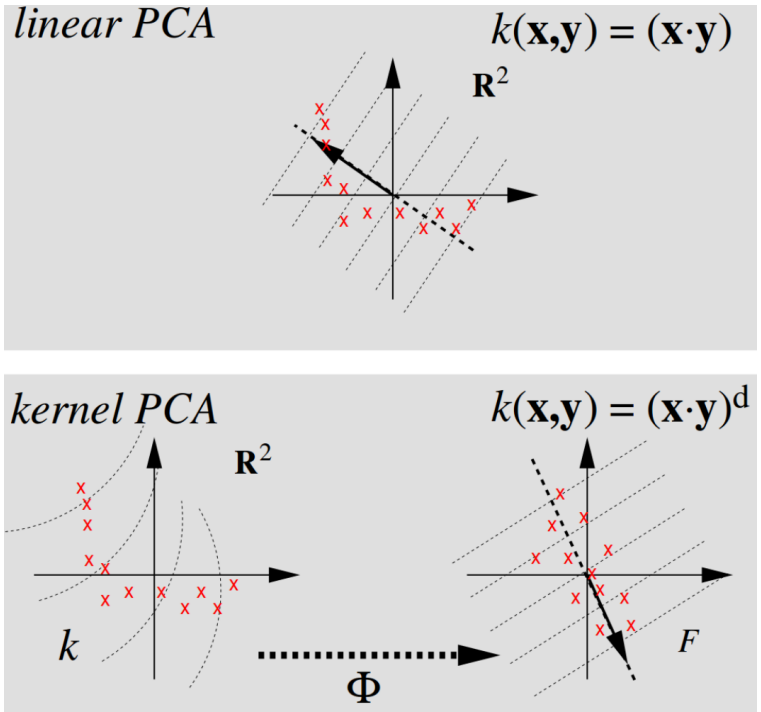


Role of dimensionality? Role of data set size?

Kernel PCA (kPCA)

Key ideas of kPCA:

- formulate all calculations of linear PCA (eigenvalue problem!) via dot products of the input space $\mathcal{X}$.
- obtain a non-linear version of linear PCA by mappings ϕ from input space $\mathcal{X}$ to a high-dimensional feature space $\mathcal{H}$!
- use the kernel trick, thus compute everything in input space $\mathcal{X}$!



Kernel PCA (kPCA)

kPCA is a non-linear feature extractor. It computes n nonlinear feature functions

$$f_1(x) = \sum_{i=1}^{m} \alpha_i^1 \; k(x_i, x)\,, \quad \ldots \quad , f_n(x) = \sum_{i=1}^{m} \alpha_i^n \; k(x_i, x)$$

where α_i^n are (up to a normalizing constant) the components of the n th eigenvector of the kernel matrix $K_{ij} := (k(x_i, x_j))$, with $n = 1, \ldots, m$

Observations:

- All mathematical and statistical properties of PCA carry over to kPCA (with the modification, that they become statements about a set of data patterns in $\mathcal{H}$ instead of vectors in $\mathcal{X}$.
- The number of feature functions is determined by the number of patterns (NOT by dimensionality of input space!)
- Computation of the Gram matrix $K_{ij} := k(x_i, x_j)$ is expensive for large number of patterns m ($\rightarrow$ iterative solutions!).

Example of kPCA

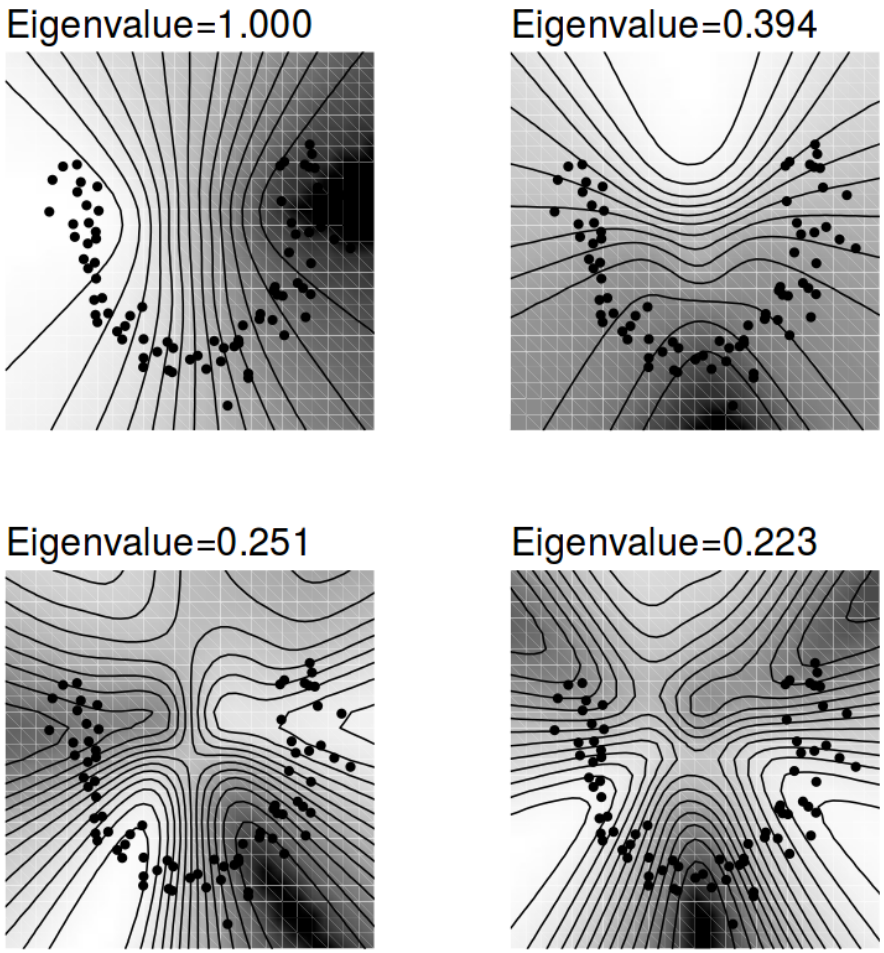


Fig. 10. The first 4 nonlinear features of Kernel-PCA using a sigmoidal Kernel on the data set from Figure 9. The Kernel-PCA components capture the nonlinear structure in the data, e.g. the first feature (upper left) is better adapted to the curvature of the data than the respective linear feature from Figure 9 (figure

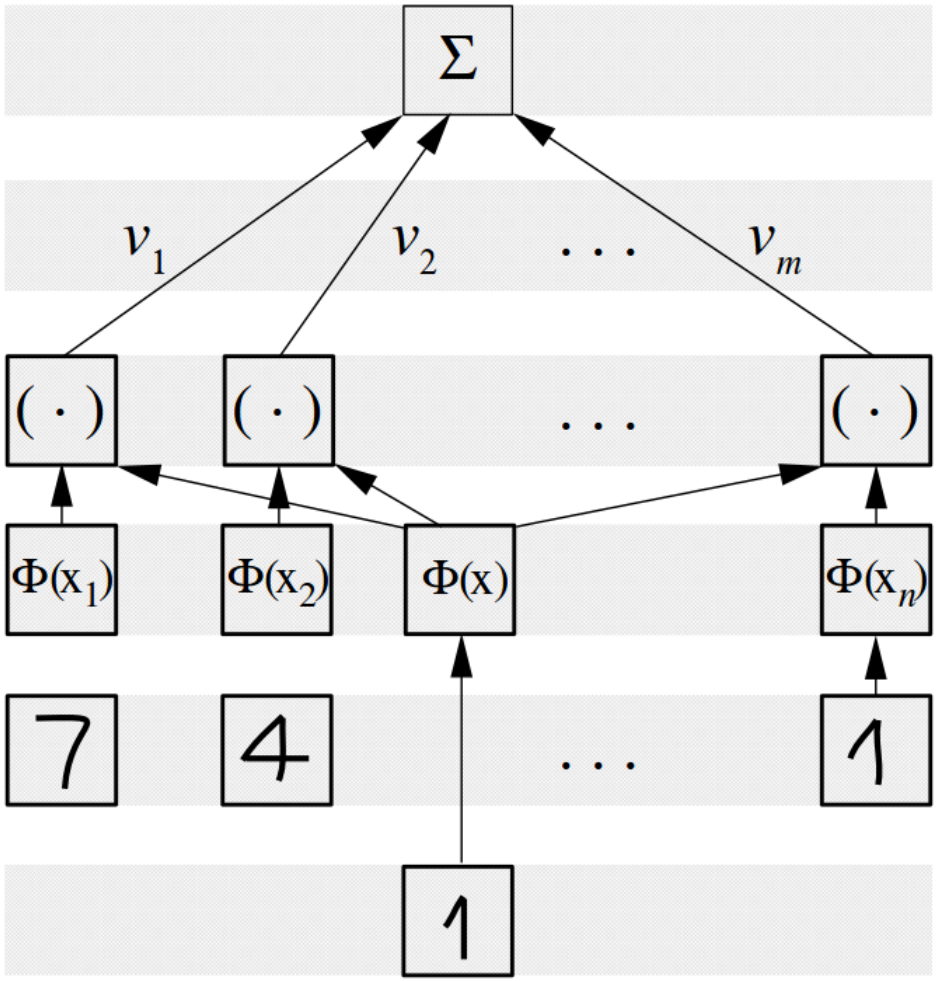
How is kPCA Applied?

- As **preprocessing** step for non-kernel methods (e.g. clustering).
- To derive **introspection** into an SVM solution (posthoc visualization of variance isolines of feature functions in $\mathcal{X}$ using the SVM hyperparameters).
- To understand, why a hard classification problem is actually hard (see "relevant dimensionality estimation" (RDE) method by Braun et al., JMLR 2008):
  1. Reason 1: the problem is intrinsically complex / high dimensional
  2. Reason 2: the problem is intrinsically simple, but data is very noisy

Further reading on kPCA: Sec. 14 of "Learning with Kernels".

Lecture Overview

Support Vector Models Can Be Expressed as Neural Networks



output $\Sigma\, v_i\, k\,(\mathrm{x},\mathrm{x}_i) + b$

weights

dot product $(\Phi(\mathrm{x})\cdot\Phi(\mathrm{x}_i)) = k(\mathrm{x},\mathrm{x}_i)$

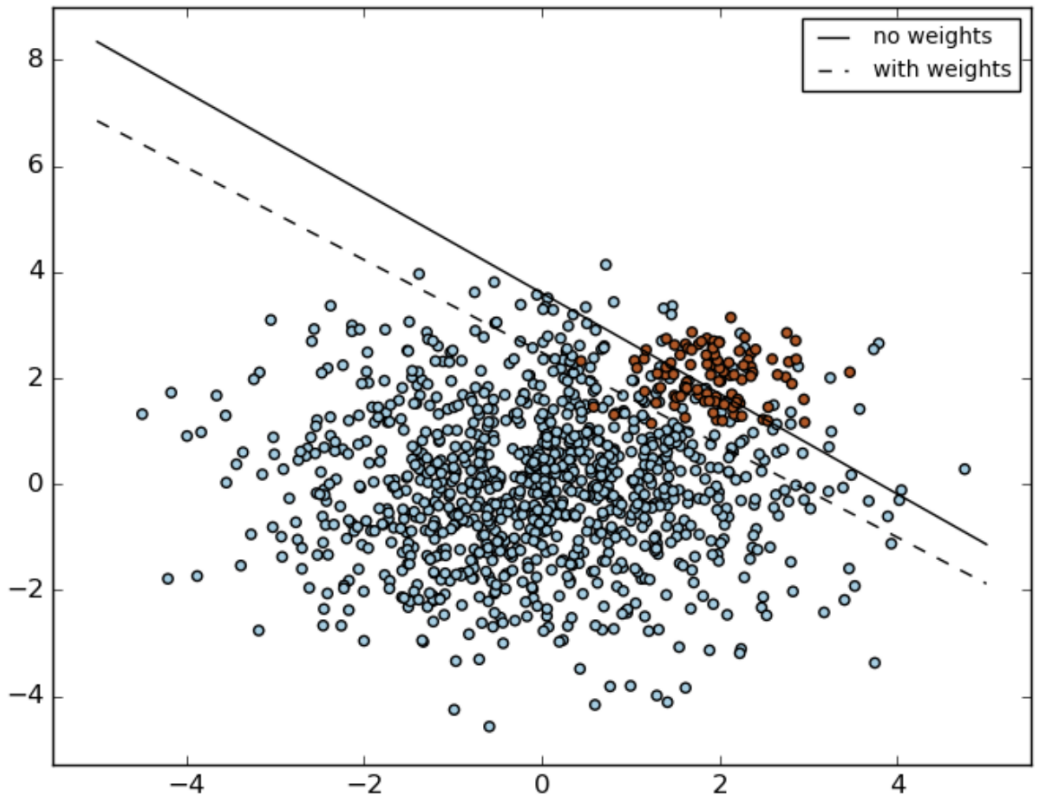
mapped vectors $\Phi(\mathrm{x}_i)$, $\Phi(\mathrm{x})$

support vectors x_1 ... x_n

test vector x

Dealing With Unbalanced Classes in SVM Training

Some classification problems have very unbalanced numbers of training patterns. By weighting the influence of misclassifications (via the slack variables!) separately per class[[[, very unbalanced problems become tractable.



Wrap-Up: Summary by Learning Goals

Having heard this lecture and done the last assignment,
you will be able to:

- formulate the optimization problems for
  - optimal large-margin hyperplane classifiers
  - (hard margin) SVM
  - soft-margin SVM
- obtain the Lagrange formulations
- explain how to get from the primal to the dual formulations
- motivate the use of slack variables, interpret support vectors
- explain (top-down) the idea of support vector regression and of kernel PCA