Advanced Machine Learning
Lecture 18
Support Vector Machines

14.01.2013

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de/
leibe@vision.rwth-aachen.de
This Lecture: **Advanced Machine Learning**

- **Regression Approaches**
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Kernels (Kernel Ridge Regression)
  - Gaussian Processes

- **Bayesian Estimation & Bayesian Non-Parametrics**
  - Prob. Distributions, Approx. Inference
  - Mixture Models & EM
  - Dirichlet Processes
  - Latent Factor Models
  - Beta Processes

- **SVMs and Structured Output Learning**
  - SVMs, SVDD, SV Regression
  - Large-margin Learning

B. Leibe
Topics of This Lecture

• Application: Nonparametric Hidden Markov Models
  - Graphical Model view
  - HDP-HMM
  - BP-HMM

• Recap: Support Vector Machines
  - Motivation
  - Primal form
  - Dual form
  - Slack variables
  - Non-linear SVMs
  - Discussion & Analysis

• Other Kernel Methods
  - Kernel PCA
  - Kernel k-Means Clustering
Hidden Markov Models (HMMs)

- Probabilistic model for sequential data
  - Widely used in speech recognition, natural language modeling, handwriting recognition, financial forecasting, ...

- Traditional view:
  - Finite state machine
  - Elements:
    - State transition matrix $A$,
    - Production probabilities $p(x \mid k)$.

- Graphical model view
  - Dynamic latent variable model
  - Elements:
    - Observation at time $n$: $x_n$
    - Hidden state at time $n$: $z_n$
    - Conditionals $p(z_{n+1} \mid z_n), p(x_n \mid z_n)$
Hidden Markov Models (HMMs)

• Traditional HMM learning
  ➢ Each state has a distribution over observable outputs \( p(x \mid k) \), e.g., modeled as a Gaussian.
  ➢ Learn the output distributions together with the transition probabilities using an EM algorithm.

• Graphical Model view
  ➢ Treat the HMM as a mixture model
  ➢ Each state is a component (“mode”) in the mixture distribution.
  ➢ From time step to time step, the responsible component switches according to the transition model.
  ➢ Advantage: we can introduce priors!
HMM: Mixture Model View

\[ z_t \sim \pi_{z_{t-1}} \]
\[ y_t \sim F(\theta_{z_t}) \]

Transition matrix
\[
P = \begin{bmatrix}
\pi_1 \\
\pi_2 \\
\vdots \\
\pi_K
\end{bmatrix}
\]

Multinomial e.g., Gaussian

Time

\[
\begin{array}{cccccc}
1 & 2 & 3 & \cdots \\
\hline
1 & & & & & \\
2 & & & & & \\
3 & & & & & \\
\vdots & & & & & \\
K & & & & & \\
\end{array}
\]

Mode

Slide credit: Erik Sudderth
HMM: Mixture Model View

\[ \alpha \rightarrow \pi_j \rightarrow \theta_k \rightarrow y_1, y_2, y_3, \ldots, y_T \]

\[ \lambda \rightarrow \theta_k \rightarrow y_1, y_2, y_3, \ldots, y_T \]

\[ \pi_2, \pi_21, \pi_22, \pi_23, \ldots, \pi_2K \]

\[ \pi_j \rightarrow \alpha \rightarrow \theta_k \rightarrow y_1, y_2, y_3, \ldots, y_T \]

**Slide credit:** Erik Sudderth
HMM: Mixture Model View

\[
\begin{align*}
\alpha & \rightarrow \pi_j_K \\
\lambda & \rightarrow \theta_K \\
& \quad \xrightarrow{z_1} 1 \quad \xrightarrow{z_3} \cdots \xrightarrow{z_T} \text{modes} \\
& \quad \xrightarrow{y_1} y_2 \quad \xrightarrow{y_3} \cdots \xrightarrow{y_T} \text{observations} \\
& \quad \xrightarrow{\pi_1} \pi_{12} \quad \pi_{13} \quad \pi_{1K} \\
& \quad \xrightarrow{1} 2 \quad 3 \quad \cdots \quad K \\
\end{align*}
\]

Time

\[
\begin{array}{cccc}
1 & 2 & 3 & \cdots \\
\Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow \\
1 & 2 & 3 & \cdots \\
\vdots & \vdots & \vdots & \vdots \\
K & 1 & 2 & 3 & \cdots \\
\end{array}
\]
HMM: Mixture Model View

**Important issue: How many modes?**

Slide credit: Erik Sudderth
Hierarchical Dirichlet Process HMM

- **Dirichlet Process**
  - Mode space of unbounded size
  - Model complexity adapts to observations

- **Hierarchical DP**
  - Ties mode transition distributions
  - *Shared* sparsity between states

Slide credit: Erik Sudderth
Beta Process HMM

- **Goal:** Transfer knowledge between related time series
  - E.g., activity recognition in video collections
  - Allow each system to switch between an arbitrarily large set of dynamical modes (“behaviors”).
  - Share behaviors across sequences.

- **Beta Processes enforce sparsity**
  - HDPs would force all videos to have non-zero probability of displaying all behaviors.
  - Beta Processes allow a video to contain only a sparse subset of relevant behaviors.

---

[B. Leibe](http://bjoernleibe.net)  
Image source: Erik Sudderth  
[Hughes & Sudderth, 2012]
Unsupervised Discovery of Activity Patterns

CMU Kitchen dataset

CMU Kitchen dataset

Image source: Erik Sudderth

Slide credit: Erik Sudderth
References and Further Reading

- **Infinite HMMs**
  - HDP-HMM
  - BP-HMMs for discovery of activity patterns
Topics of This Lecture

- **Application: Nonparametric Hidden Markov Models**
  - Graphical Model view
  - HDP-HMM
  - BP-HMM

- **Recap: Support Vector Machines**
  - Motivation
  - Primal form
  - Dual form
  - Slack variables
  - Non-linear SVMs
  - Discussion & Analysis

- **Other Kernel Methods**
  - Kernel PCA
  - Kernel k-Means Clustering
Recap: Support Vector Machine (SVM)

• Basic idea
  - The SVM tries to find a classifier which maximizes the margin between pos. and neg. data points.
  - Up to now: consider linear classifiers

\[ w^T x + b = 0 \]

• Formulation as a convex optimization problem
  - Find the hyperplane satisfying

\[ \arg\min_{w,b} \frac{1}{2} \|w\|^2 \]

under the constraints

\[ t_n (w^T x_n + b) \geq 1 \quad \forall n \]

based on training data points \( x_n \) and target values \( t_n \in \{-1, 1\} \).
Recap: SVM - Lagrangian Formulation

- Find hyperplane minimizing $\|w\|^2$ under the constraints
  $$t_n(w^T x_n + b) - 1 \geq 0 \quad \forall n$$

- Lagrangian formulation
  - Introduce positive Lagrange multipliers: $a_n \geq 0 \quad \forall n$
  - Minimize Lagrangian ("primal form")
    $$L(w, b, a) = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \{t_n(w^T x_n + b) - 1\}$$
    - i.e., find $w$, $b$, and $a$ such that
      $$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} a_n t_n = 0$$
      $$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{n=1}^{N} a_n t_n x_n$$
Recap: SVM - Primal Formulation

- Lagrangian primal form
  \[ L_p = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n (w^T x_n + b) - 1 \right\} \]
  \[ = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n y(x_n) - 1 \right\} \]

- The solution of \( L_p \) needs to fulfill the KKT conditions
  - Necessary and sufficient conditions
    \[ a_n \geq 0 \]
    \[ t_n y(x_n) - 1 \geq 0 \]
    \[ a_n \left\{ t_n y(x_n) - 1 \right\} = 0 \]

\[ \text{KKT:} \]
\[ \lambda \geq 0 \]
\[ f(x) \geq 0 \]
\[ \lambda f(x) = 0 \]
Recap: SVM - Solution

- Solution for the hyperplane
  - Computed as a linear combination of the training examples
    \[ w = \sum_{n=1}^{N} a_n t_n x_n \]
  - Sparse solution: \( a_n \neq 0 \) only for some points, the support vectors
    \( \Rightarrow \) Only the SVs actually influence the decision boundary!
  - Compute \( b \) by averaging over all support vectors:
    \[ b = \frac{1}{NS} \sum_{n \in S} \left( t_n - \sum_{m \in S} a_m t_m x_m^T x_n \right) \]
Recap: SVM - Support Vectors

• The training points for which $a_n > 0$ are called "support vectors".

• Graphical interpretation:
  - The support vectors are the points on the margin.
  - They define the margin and thus the hyperplane.

⇒ All other data points can be discarded!
Recap: SVM - Dual Formulation

• Improving the scaling behavior: rewrite $L_p$ in a dual form

$$L_p = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \{ t_n (w^T x_n + b) - 1 \}$$

$$= \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n t_n w^T x_n - b \sum_{n=1}^{N} a_n t_n + \sum_{n=1}^{N} a_n$$

> Using the constraint $\sum_{n=1}^{N} a_n t_n = 0$, we obtain

$$\frac{\partial L_p}{\partial b} = 0$$

$$L_p = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n t_n w^T x_n + \sum_{n=1}^{N} a_n$$
Recap: SVM - Dual Formulation

\[ L_p = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n t_n w^T x_n + \sum_{n=1}^{N} a_n \]

- Using the constraint \( w = \sum_{n=1}^{N} a_n t_n x_n \), we obtain

\[ \frac{\partial L_p}{\partial w} = 0 \]

\[ L_p = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n t_n \sum_{m=1}^{N} a_m t_m x_m^T x_n + \sum_{n=1}^{N} a_n \]

\[ = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n) + \sum_{n=1}^{N} a_n \]
Recap: SVM - Dual Formulation

\[ L = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n) + \sum_{n=1}^{N} a_n \]

- Applying \( \frac{1}{2} \|w\|^2 = \frac{1}{2} w^T w \) and again using \( w = \sum_{n=1}^{N} a_n t_n x_n \)

\[ \frac{1}{2} w^T w = \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n) \]

- Inserting this, we get the Wolfe dual

\[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n) \]
Recap: SVM - Dual Formulation

- Maximize

\[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n) \]

under the conditions

\[ a_n \geq 0 \quad \forall n \]

\[ \sum_{n=1}^{N} a_n t_n = 0 \]

- Comparison

- \( L_d \) is equivalent to the primal form \( L_p \), but only depends on \( a_n \).
- \( L_p \) scales with \( \mathcal{O}(D^3) \).
- \( L_d \) scales with \( \mathcal{O}(N^3) \) - in practice between \( \mathcal{O}(N) \) and \( \mathcal{O}(N^2) \).

Slide adapted from Bernt Schiele
Recap: SVM for Non-Separable Data

- Slack variables
  - One slack variable $\xi_n \geq 0$ for each training data point.

- Interpretation
  - $\xi_n = 0$ for points that are on the correct side of the margin.
  - $\xi_n = |t_n - y(x_n)|$ for all other points.

We do not have to set the slack variables ourselves!
⇒ They are jointly optimized together with $w$. 

Point on decision boundary: $\xi_n = 1$

Misclassified point: $\xi_n > 1$
Recap: SVM - Non-Separable Data

- **Separable data**
  - Minimize $$\frac{1}{2} \|w\|^2$$

- **Non-separable data**
  - Minimize $$\frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n$$

Trade-off parameter!
Recap: SVM - New Primal Formulation

- **New SVM Primal: Optimize**

\[
L_p = \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} a_n (t_n y(x_n) - 1 + \xi_n) - \sum_{n=1}^{N} \mu_n \xi_n
\]

Constraint: \( t_n y(x_n) \geq 1 - \xi_n \)

Constraint: \( \xi_n \geq 0 \)

- **KKT conditions**

\[
a_n \geq 0 \quad \mu_n \geq 0
\]

\[
t_n y(x_n) - 1 + \xi_n \geq 0 \quad \xi_n \geq 0
\]

\[
a_n (t_n y(x_n) - 1 + \xi_n) = 0 \quad \mu_n \xi_n = 0
\]

**KKT:**

\[
\lambda \geq 0
\]

\[
f(x) \geq 0
\]

\[
\lambda f(x) = 0
\]
Recap: SVM - New Dual Formulation

- **New SVM Dual: Maximize**

  \[
  L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n)
  \]

  under the conditions

  \[
  0 \cdot a_n \cdot C
  \]

  \[
  \sum_{n=1}^{N} a_n t_n = 0
  \]

- **This is again a quadratic programming problem**

  \(\Rightarrow\) Solve as before...

This is all that changed!
Recap: Nonlinear SVMs

- General idea: The original input space can be mapped to some higher-dimensional feature space where the training set is separable:

\[
\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})
\]
Recap: The Kernel Trick

- Important observation
  - $\phi(x)$ only appears in the form of dot products $\phi(x)^T \phi(y)$:
    $$y(x) = w^T \phi(x) + b$$
    $$= \sum_{n=1}^{N} a_n t_n \phi(x_n)^T \phi(x) + b$$
  - Define a so-called kernel function $k(x,y) = \phi(x)^T \phi(y)$.
  - Now, in place of the dot product, use the kernel instead:
    $$y(x) = \sum_{n=1}^{N} a_n t_n k(x_n, x) + b$$
  - The kernel function \textit{implicitly} maps the data to the higher-dimensional space (without having to compute $\phi(x)$ explicitly)!
Recap: SVMs with Kernels

• Using kernels
  ➢ Applying the kernel trick is easy. Just replace every dot product by a kernel function...
    \[ x^T y \rightarrow k(x, y) \]
  ➢ ...and we’re done.
  ➢ Instead of the raw input space, we’re now working in a higher-dimensional (potentially infinite dimensional!) space, where the data is more easily separable.

  “Sounds like magic...”

• Wait - does this always work?
  ➢ The kernel needs to define an implicit mapping to a higher-dimensional feature space \( \phi(x) \).
  ➢ Kernel needs to fulfill Mercer’s condition (→ Lecture 4).
Recap: Nonlinear SVM - Dual Formulation

- **SVM Dual: Maximize**

\[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(x_m, x_n) \]

under the conditions

\[ 0 \cdot a_n \cdot C \]

\[ \sum_{n=1}^{N} a_n t_n = 0 \]

- **Classify new data points using**

\[ y(x) = \sum_{n=1}^{N} a_n t_n k(x_n, x) + b \]
Summary: SVMs

• Properties
  - Empirically, SVMs work very, very well.
  - SVMs are currently among the best performers for a number of classification tasks ranging from text to genomic data.
  - SVMs can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.
  - SVM techniques have been applied to a variety of other tasks
    - e.g. SV Regression, One-class SVMs, ...
  - The kernel trick has been used for a wide variety of applications. It can be applied wherever dot products are in use
    - e.g. Kernel PCA, kernel FLD, ...
    - Good overview, software, and tutorials available on http://www.kernel-machines.org/
You Can Try It At Home...

- Lots of SVM software available, e.g.
    - Command-line based interface
    - Source code available (in C)
    - Interfaces to Python, MATLAB, Perl, Java, DLL,...
    - Library for inclusion with own code
    - C++ and Java sources
    - Interfaces to Python, R, MATLAB, Perl, Ruby, Weka, C+ .NET,...
  - Both include fast training and evaluation algorithms, support for multi-class SVMs, automated training and cross-validation, ...
    ⇒ Easy to apply to your own problems!
Topics of This Lecture

• Application: Nonparametric Hidden Markov Models
  - Graphical Model view
  - HDP-HMM
  - BP-HMM

• Recap: Support Vector Machines
  - Motivation
  - Primal form
  - Dual form
  - Slack variables
  - Non-linear SVMs
  - Discussion & Analysis

• Other Kernel Methods
  - Kernel PCA
  - Kernel k-Means Clustering
SVM - Analysis

- Traditional soft-margin formulation

\[
\min_{w \in \mathbb{R}^D, \xi_n \in \mathbb{R}^+} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n
\]

subject to the constraints

\[
t_n y(x_n) \geq 1 - \xi_n
\]

“Maximize the margin”

“Most points should be on the correct side of the margin”

- Different way of looking at it

  - We can reformulate the constraints into the objective function.

\[
\min_{w \in \mathbb{R}^D} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \left[1 - t_n y(x_n)\right]_+
\]

L₂ regularizer  “Hinge loss”

where \([x]_+ := \max\{0,x\}\).
Error Functions (Loss Functions)

- Ideal misclassification error function (black)
  - This is what we want to approximate.
  - Unfortunately, it is not differentiable.
  - The gradient is zero for misclassified points.
  $\Rightarrow$ We cannot minimize it by gradient descent.

$$z_n = t_n y(x_n)$$

Image source: Bishop, 2006
Error Functions (Loss Functions)

- **Squared error used in Least-Squares Classification**
  - Very popular, leads to closed-form solutions.
  - However, sensitive to outliers due to squared penalty.
  - Penalizes “too correct” data points
  ⇒ Generally does not lead to good classifiers.

Sensitivity to outliers!

Ideal misclassification error
Squared error

Penalizes “too correct” data points!
Error Functions (Loss Functions)

- “Hinge error” used in SVMs
  - Zero error for points outside the margin ($z_n > 1$).
  - Linearly increasing error for misclassified points ($z_n < 1$).
  $\Rightarrow$ Leads to sparse solutions, not sensitive to outliers.
  - Not differentiable around $z_n = 1 \Rightarrow$ Cannot be optimized directly.

Ideal misclassification error
Squared error
Hinge error

Robust to outliers!
Not differentiable!
Favors sparse solutions!

Image source: Bishop, 2006
SVM - Discussion

- SVM optimization function

\[
\min_{\mathbf{w} \in \mathbb{R}^D} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^{N} [1 - t_n y(x_n)]_+
\]

- Hinge loss enforces sparsity
  - Only a subset of training data points actually influences the decision boundary.
  - This is different from sparsity obtained through the regularizer! There, only a subset of input dimensions are used.
  - Unconstrained optimization, but non-differentiable function.
  - Solve, e.g. by subgradient descent
  - Currently most efficient: stochastic gradient descent

Slide adapted from Christoph Lampert
Outline of the Remaining Lectures

- **We will generalize the SVM idea in several directions...**

- Other Kernel methods
  - Kernel PCA
  - Kernel k-Means

- Other Large-Margin Learning formulations
  - Support Vector Data Description (one-class SVMs)
  - Support Vector Regression

- Structured Output Learning
  - General loss functions
  - General structured outputs
  - Structured Output SVM
  - Example: Multiclass SVM
Topics of This Lecture

• Application: Nonparametric Hidden Markov Models
  - Graphical Model view
  - HDP-HMM
  - BP-HMM

• Recap: Support Vector Machines
  - Motivation
  - Primal form
  - Dual form
  - Slack variables
  - Non-linear SVMs
  - Discussion & Analysis

• Other Kernel Methods
  - Kernel PCA
  - Kernel k-Means Clustering
Recap: PCA

- PCA procedure
  - Given samples $x_n \in \mathbb{R}^d$, PCA finds the directions of maximal covariance. Without loss of generality assume that $\sum_n x_n = 0$.
  - The PCA directions $e_1, \ldots, e_d$ are the eigenvectors of the covariance matrix
    \[
    C = \frac{1}{N} \sum_{n=1}^{N} x_n x_n^T
    \]
    sorted by their eigenvalue.
  - We can express $x_n$ in PCA space by
    \[
    F(x_n) = \sum_{k=1}^{K} \langle x_n, e_k \rangle e_k
    \]
  - Lower-dim. coordinate mapping: $x_n \mapsto \left( \begin{array}{c} \langle x_n, e_1 \rangle \\ \langle x_n, e_2 \rangle \\ \vdots \\ \langle x_n, e_K \rangle \end{array} \right) \in \mathbb{R}^K$
Kernel-PCA

- **Kernel-PCA procedure**
  - Given samples $x_n \in \mathcal{X}$, kernel $\mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ with an implicit feature map $\phi: \mathcal{X} \rightarrow \mathcal{H}$. Perform PCA in the Hilbert space $\mathcal{H}$.
  - The kernel-PCA directions $e_1, \ldots, e_d$ are the eigenvectors of the covariance operator $C = \frac{1}{N} \sum_{n=1}^{N} \phi(x_n)\phi(x_n)^T$ sorted by their eigenvalue.
  - Lower-dim. coordinate mapping: $x_n \mapsto \begin{pmatrix} \langle \phi(x_n), e_1 \rangle \\ \langle \phi(x_n), e_2 \rangle \\ \vdots \\ \langle \phi(x_n), e_K \rangle \end{pmatrix} \in \mathbb{R}^K$

Slide credit: Christoph Lampert
Kernel-PCA

- Kernel-PCA procedure
  - Given samples $x_n \in \mathcal{X}$, kernel $\mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ with an implicit feature map $\phi: \mathcal{X} \rightarrow \mathcal{H}$. Perform PCA in the Hilbert space $\mathcal{H}$.
  - Equivalently, we can use the eigenvectors $e'_k$ and eigenvalues $\lambda_k$ of the kernel matrix
    \[
    K = \left( \langle \phi(x_m), \phi(x_n) \rangle \right)_{m,n=1,...,N} = \left( k(x_m, x_n) \right)_{m,n=1,...,N}
    \]
  - Coordinate mapping:
    \[
    x_n \mapsto (\sqrt{\lambda_1}e'_1, ..., \sqrt{\lambda_K}e'_K)
    \]
Example: Image Superresolution

- Training procedure
  - Collect high-res face images
  - Use KPCA with RBF-kernel to learn non-linear subspaces

- For new low-res image:
  - Scale to target high resolution
  - Project to closest point in face subspace


Slide credit: Christoph Lampert
Kernel k-Means Clustering

- Kernel PCA is more than just non-linear versions of PCA
  - PCA maps $\mathbb{R}^d$ to $\mathbb{R}^{d'}$, e.g., to remove noise dimensions.
  - Kernel-PCA maps $\mathcal{X} \rightarrow \mathbb{R}^{d'}$, so it provides a vectorial representation of non-vectorial data.
  - We can apply algorithms that only work in vector spaces to data that is not in a vector representation.

- Example: k-Means clustering
  - Given $x_1, \ldots, x_n \in \mathcal{X}$.
  - Choose a kernel function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$.
  - Apply kernel-PCA to obtain vectorial $v_1, \ldots, v_n \in \mathbb{R}^{d'}$.
  - Cluster $v_1, \ldots, v_n \in \mathbb{R}^{d'}$ using K-Means.
  - $x_1, \ldots, x_n$ are clustered based on the similarity defined by $k$. 

Slide credit: Christoph Lampert
Example: Unsupervised Object Categorization

- Automatically group images that show similar objects
  - Represent images by bag-of-word histograms
  - Perform Kernel k-Means Clustering
  \[ \Rightarrow \text{Observation: Clusters get better if we use a good image kernel (e.g., } \chi^2) \text{ instead of plain k-Means (linear kernel).} \]


Slide credit: Christoph Lampert
References and Further Reading

• More information on SVMs can be found in Chapter 7.1 of Bishop’s book. You can also look at Schölkopf & Smola (some chapters available online).

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006

B. Schölkopf, A. Smola
Learning with Kernels
MIT Press, 2002
http://www.learning-with-kernels.org/