Advanced Machine Learning
Lecture 18

Support Vector Machines

14.01.2013

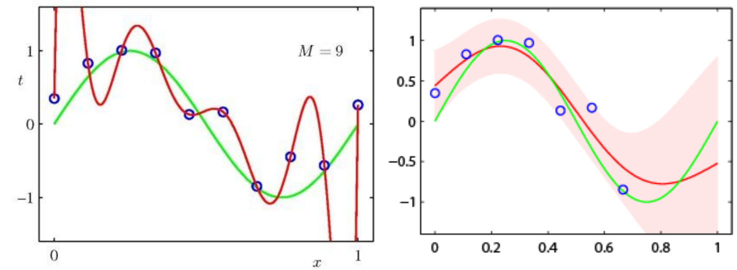
Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de/

leibe@vision.rwth-aachen.de

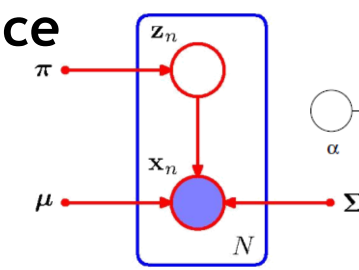
This Lecture: *Advanced Machine Learning*

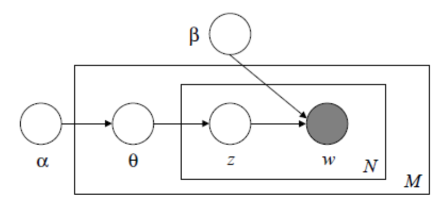
- **Regression Approaches**
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Kernels (Kernel Ridge Regression)
  - Gaussian Processes

$$f : \mathcal{X} \rightarrow \mathbb{R}$$



- **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

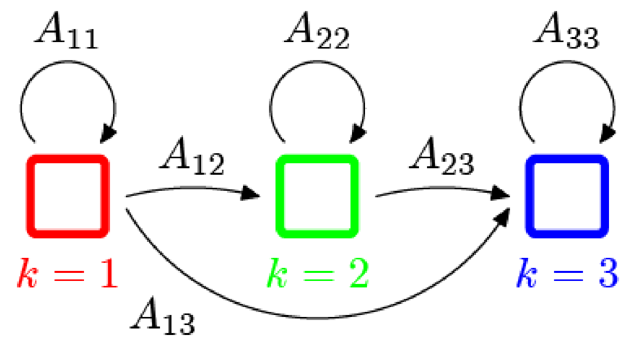
$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

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 $\mathbf{A}$,
    - Production probabilities $p(\mathbf{x} \mid k)$.



- Graphical model view
  - Dynamic latent variable model
  - Elements:
    - Observation at time n: $\mathbf{x}_n$
    - Hidden state at time n: $\mathbf{z}_n$
    - Conditionals $p(\mathbf{z}_{n+1}|\mathbf{z}_n)$, $p(\mathbf{x}_n|\mathbf{z}_n)$

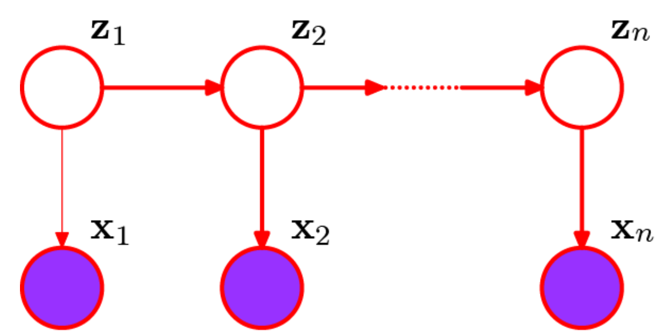
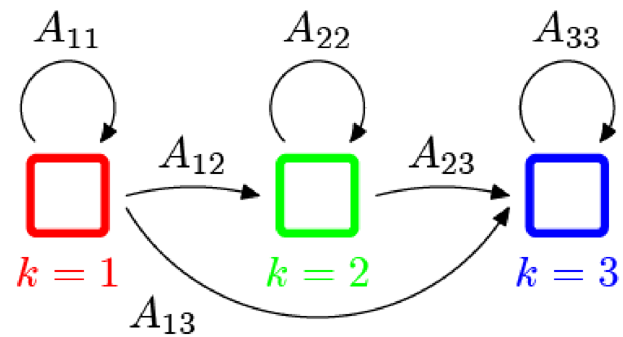


Image source: C.M. Bishop, 2006

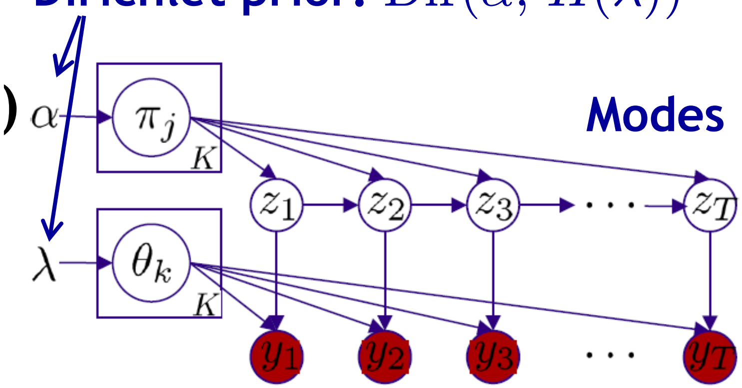
Hidden Markov Models (HMMs)

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

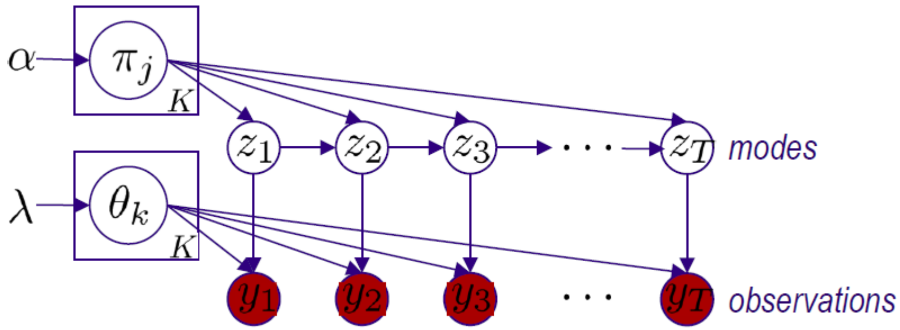


- Graphical Model view
  - Treat the HMM as a mixture model
  - Each state is a component ("mode") in the mixture distribution.
  - From time step to time step, the responsible component switches according to the transition model.
  - *Advantage: we can introduce priors!*

Dirichlet prior: $\mathrm{Dir}(\alpha, H(\lambda))$



HMM: Mixture Model View

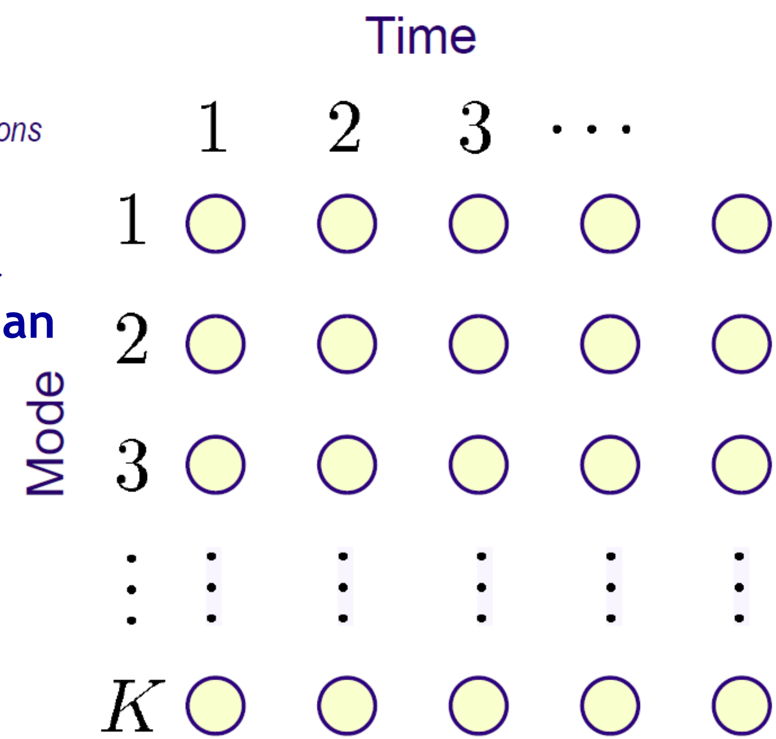


$z_t \sim \pi_{z_{t-1}}$ **Multinomial**

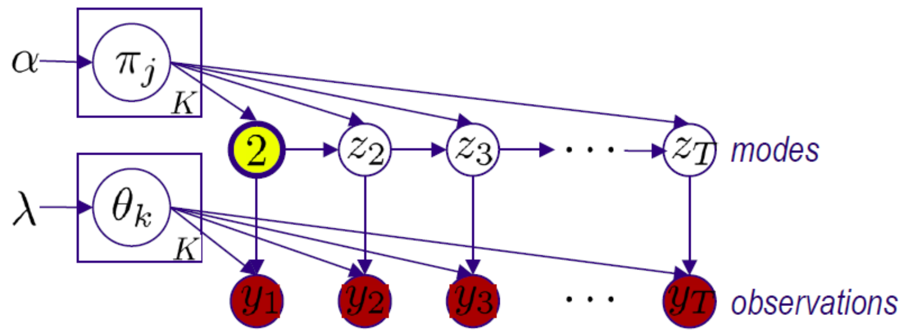
$y_t \sim F(\theta_{z_t})$ **e.g., Gaussian**

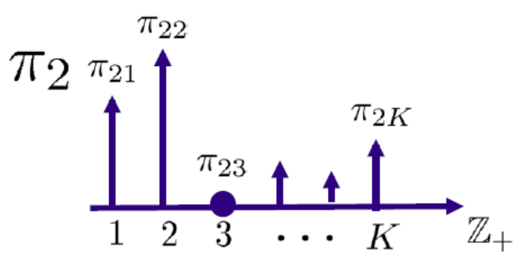
$$P = \begin{bmatrix} \text{---} & \pi_1 & \text{---} \\ \text{---} & \pi_2 & \text{---} \\ & \vdots & \\ \text{---} & \pi_K & \text{---} \end{bmatrix}$$

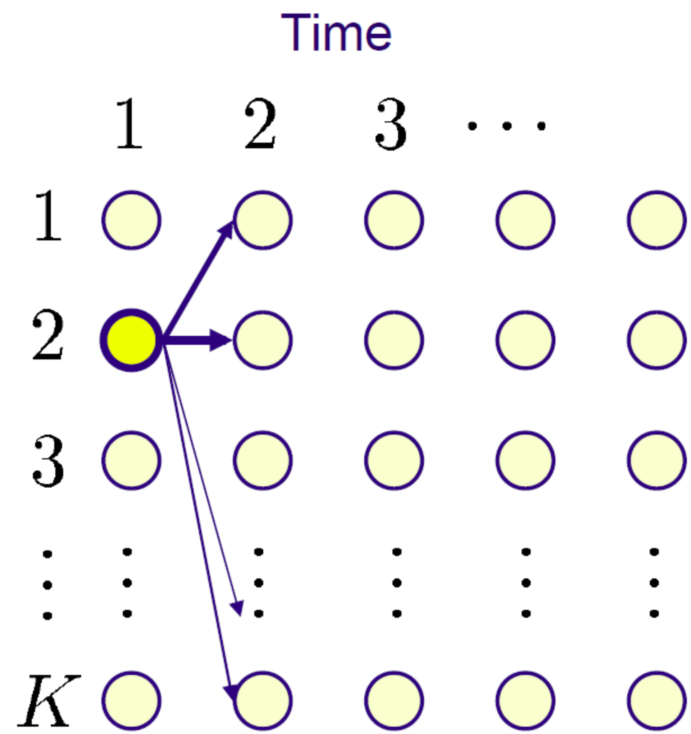
Transition matrix



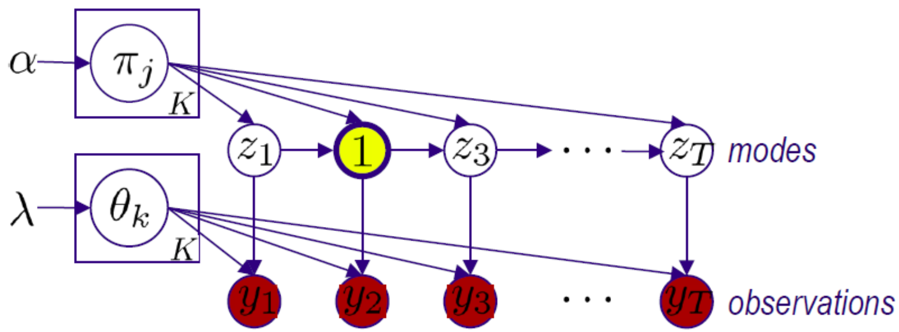
HMM: Mixture Model View

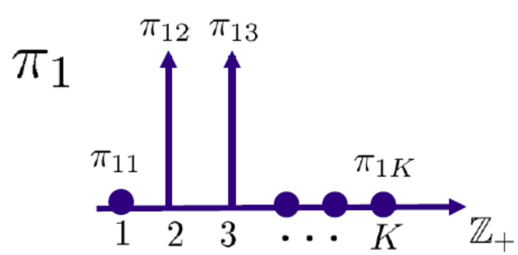


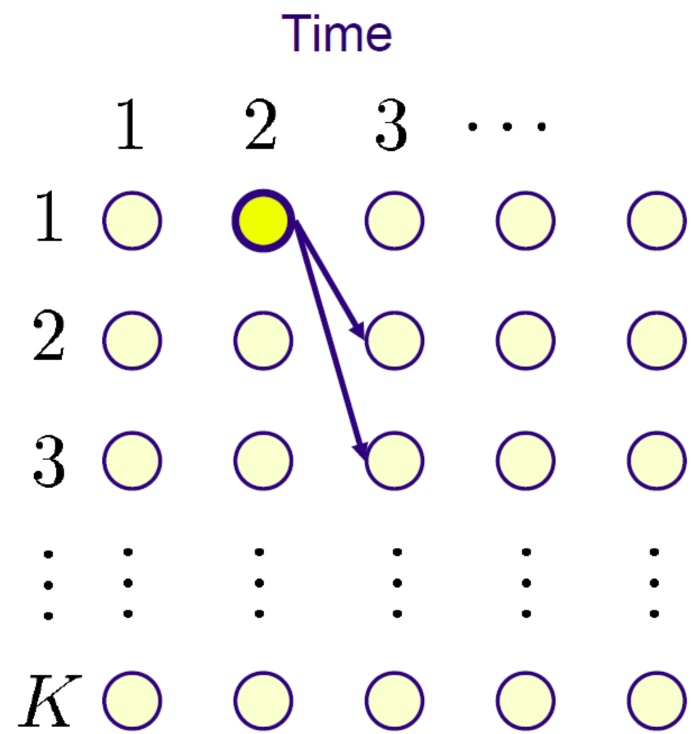




HMM: Mixture Model View

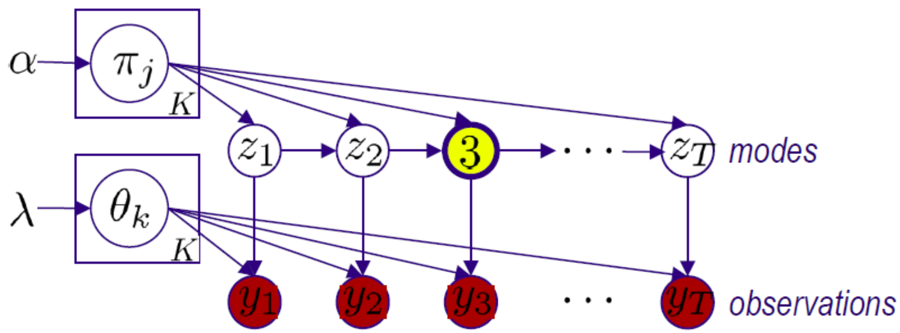


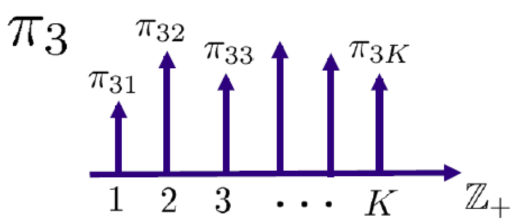


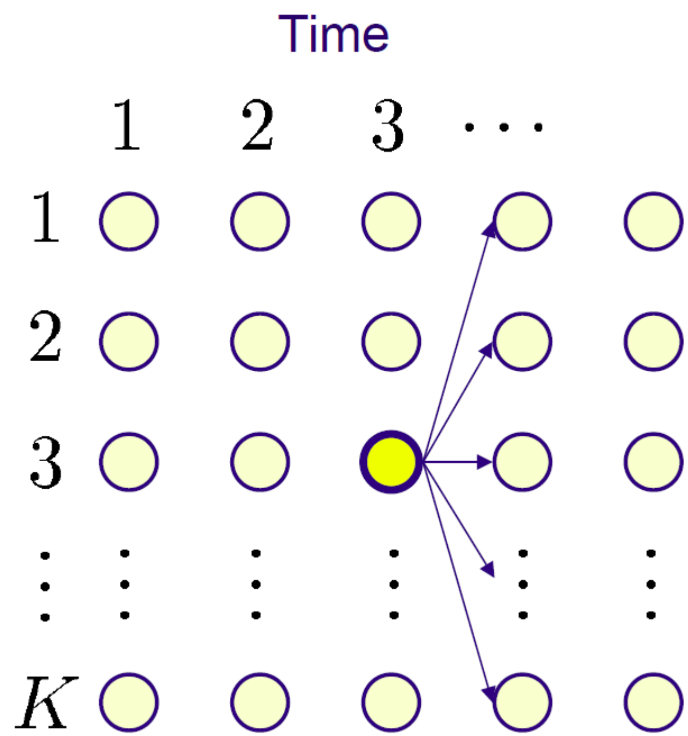


Slide credit: Erik Sudderth

HMM: Mixture Model View



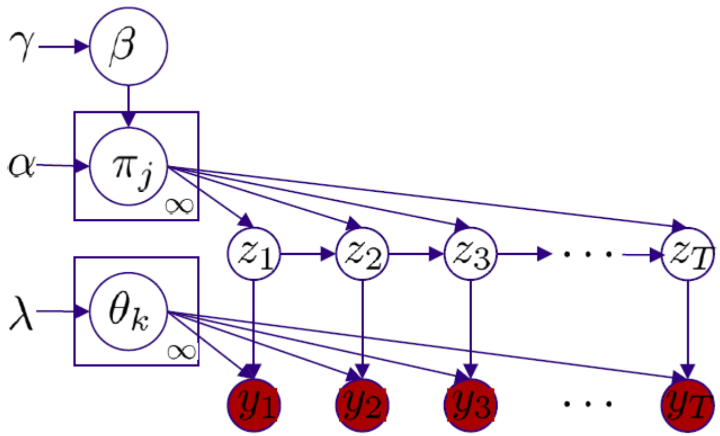




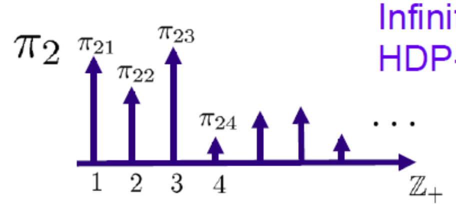
Important issue: How many modes?

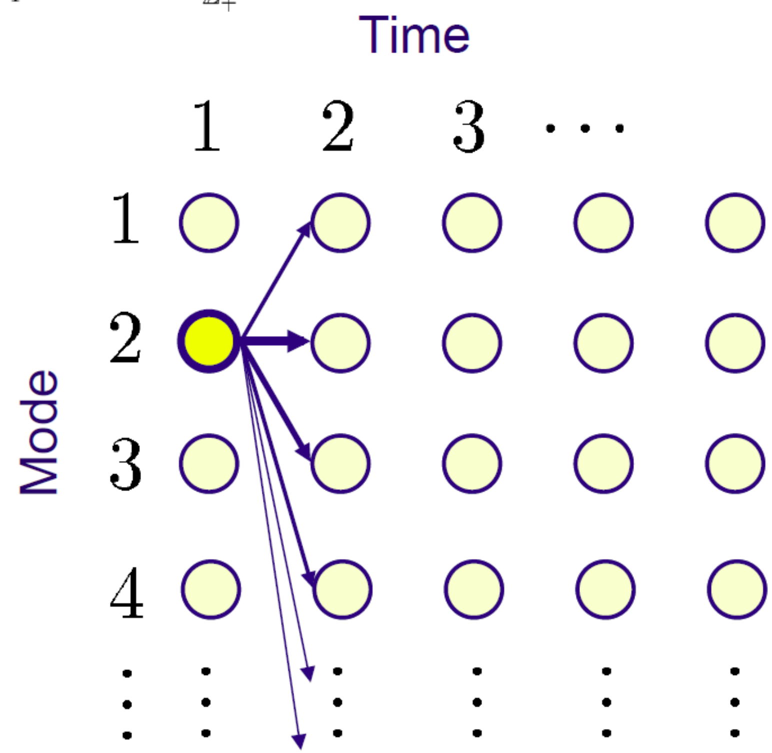
Hierarchical Dirichlet Process HMM

Infinite HMM: Beal, et.al., *NIPS* 2002
HDP-HMM: Teh, et. al., *JASA* 2006



HDP HMM



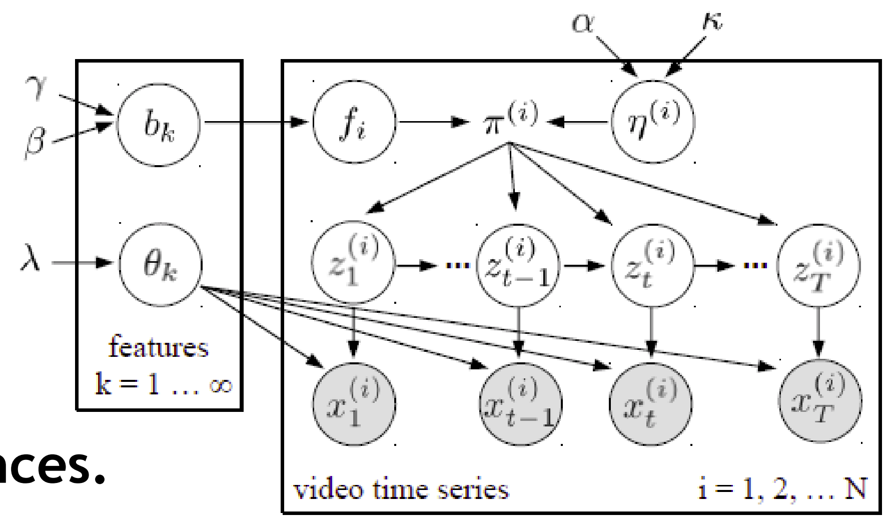


Infinite state space

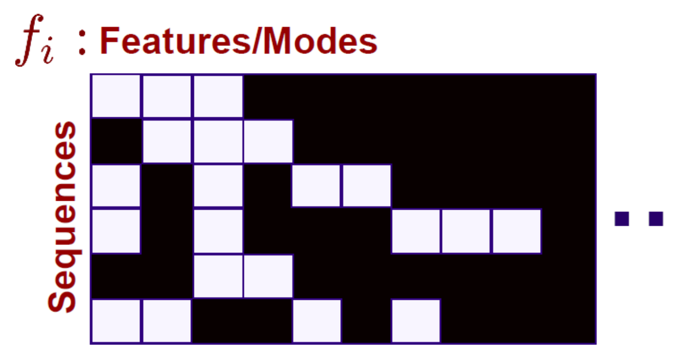
- **Dirichlet Process**
  - Mode space of unbounded size
  - Model complexity adapts to observations
- **Hierarchical DP**
  - Ties mode transition distributions
  - *Shared* sparsity between states

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.



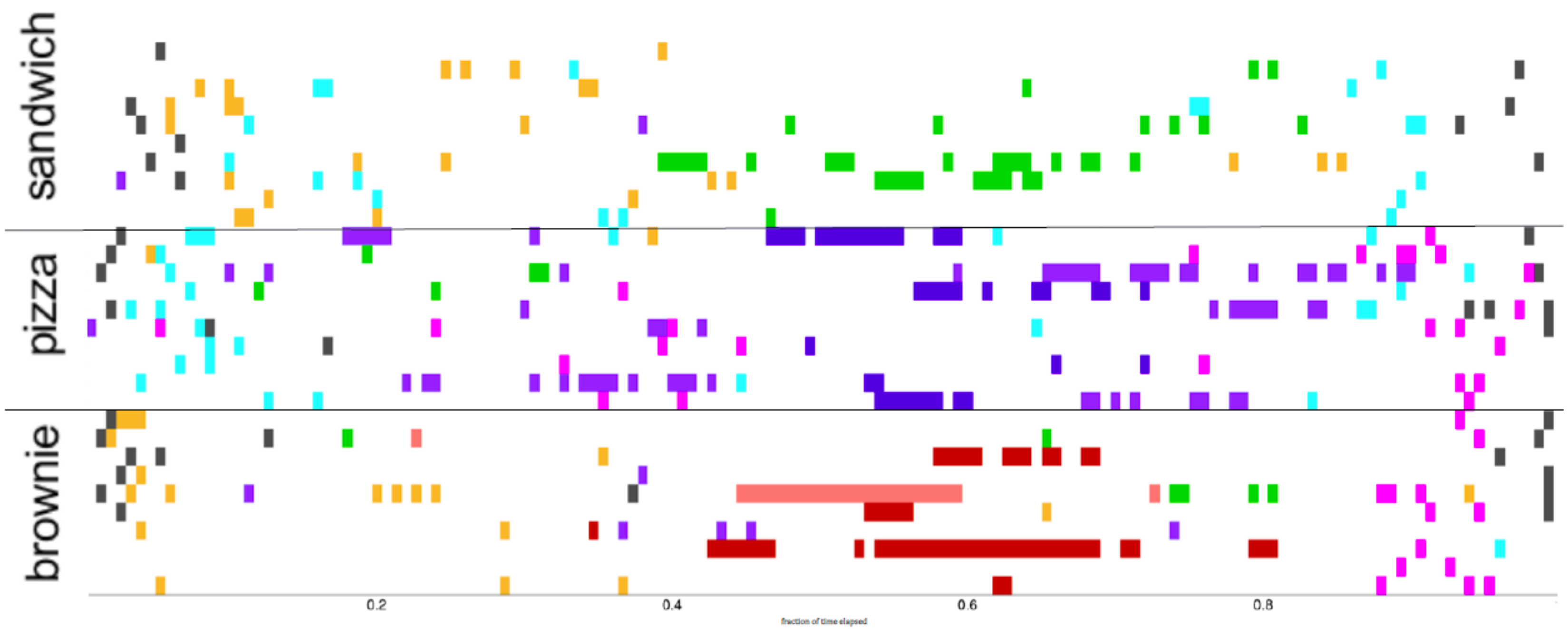
- Beta Processes enforce sparsity
  - HDPs would force all videos to have non-zero probability of displaying all behaviors.
  - Beta Processes allow a video to contain only a sparse subset of relevant behaviors.



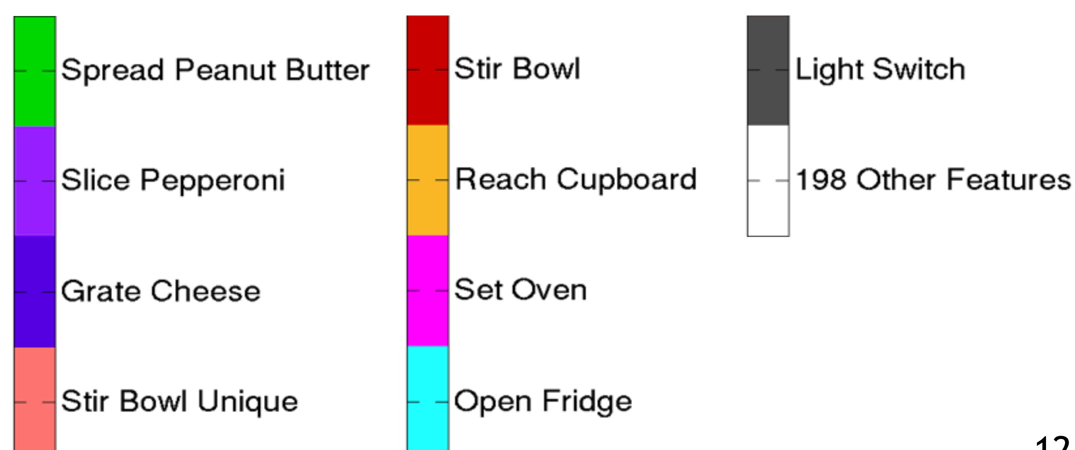
[Hughes & Sudderth, 2012]

Image source: Erik Sudderth

Unsupervised Discovery of Activity Patterns



CMU Kitchen dataset



Slide credit: Erik Sudderth

Image source: Erik Sudderth

References and Further Reading

- Infinite HMMs
  - HDP-HMM
    - J. Paisley, F. Carin, Nonparametric Factor Analysis with Beta Process Priors, ICML 2009.
  - BP-HMMs for discovery of activity patterns
    - M.C. Hughes, E.B. Sudderth, Nonparametric Discovery of Activity Patterns from Video Collections. CVPR Workshop on Perceptual Organization in Computer Vision, 2012.

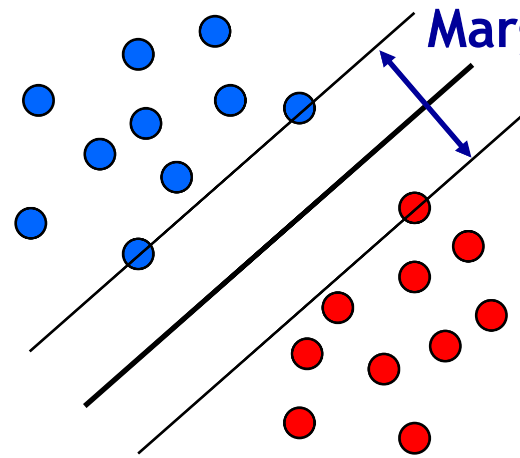
Topics of This Lecture

- 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

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$$



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

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

under the constraints

$$t_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) \geq 1 \quad \forall n$$

based on training data points $\mathbf{x}_n$ and target values $t_n \in \{-1, 1\}$.

Recap: SVM – Lagrangian Formulation

- Find hyperplane minimizing $\|\mathbf{w}\|^2$ under the constraints

$$t_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) - 1 \geq 0 \quad \forall n$$

- Lagrangian formulation
  - Introduce positive Lagrange multipliers: $a_n \geq 0 \quad \forall n$
  - Minimize Lagrangian ("primal form")

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left\{t_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) - 1\right\}$$

  - I.e., find $\mathbf{w}$, b, and $\mathbf{a}$ such that

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \boxed{\sum_{n=1}^{N} a_n t_n = 0} \qquad \frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \boxed{\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n}$$

Recap: SVM – Primal Formulation

- Lagrangian primal form

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) - 1 \right\}$$

$$= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n y(\mathbf{x}_n) - 1 \right\}$$

- The solution of L_p needs to fulfill the KKT conditions
  - Necessary and sufficient conditions

$$\begin{aligned} a_n &\geq 0 \\ t_n y(\mathbf{x}_n) - 1 &\geq 0 \\ a_n \left\{ t_n y(\mathbf{x}_n) - 1 \right\} &= 0 \end{aligned}$$

KKT:

$$\begin{aligned} \lambda &\geq 0 \\ f(\mathbf{x}) &\geq 0 \\ \lambda f(\mathbf{x}) &= 0 \end{aligned}$$

Recap: SVM – Solution

- **Solution for the hyperplane**
  - Computed as a linear combination of the training examples

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$$

  - 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:

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^{\mathrm{T}} \mathbf{x}_n \right)$$

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.

$\Rightarrow$ All other data points can be discarded!

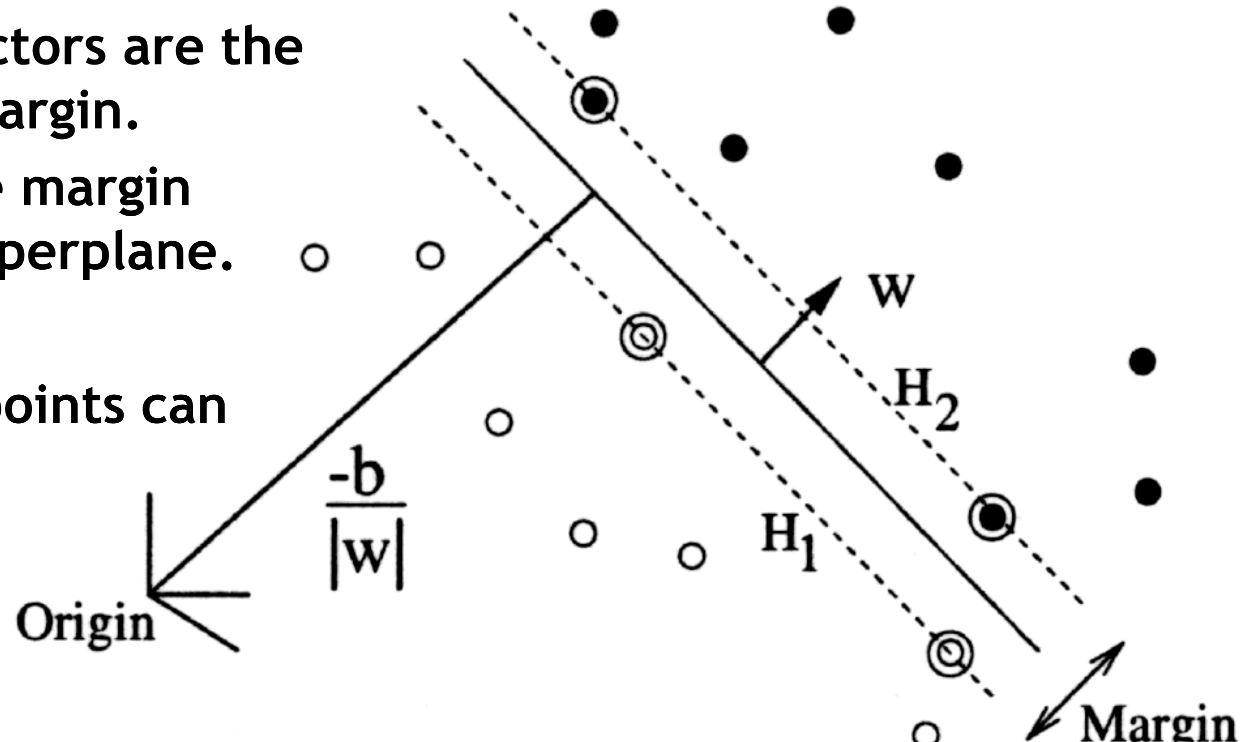


Image source: C. Burges, 1998

Recap: SVM – Dual Formulation

- **Improving the scaling behavior: rewrite L_p in a dual form**

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) - 1 \right\}$$

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

=0

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

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

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n t_n \mathbf{w}^{\mathrm{T}}\mathbf{x}_n + \sum_{n=1}^{N} a_n$$

Recap: SVM – Dual Formulation

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n t_n \mathbf{w}^{\mathrm{T}} \mathbf{x}_n + \sum_{n=1}^{N} a_n$$

- **Using the constraint** $\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$ **, we obtain**

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0$$

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n t_n \sum_{m=1}^{N} a_m t_m \mathbf{x}_m^{\mathrm{T}} \mathbf{x}_n + \sum_{n=1}^{N} a_n$$

$$= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (\mathbf{x}_m^{\mathrm{T}} \mathbf{x}_n) + \sum_{n=1}^{N} a_n$$

Recap: SVM – Dual Formulation

$$L = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m (\mathbf{x}_m^{\mathrm{T}}\mathbf{x}_n) + \sum_{n=1}^{N} a_n$$

- Applying $\frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$ and again using $\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$

$$\frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m (\mathbf{x}_m^{\mathrm{T}}\mathbf{x}_n)$$

- Inserting this, we get the Wolfe dual

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m (\mathbf{x}_m^{\mathrm{T}}\mathbf{x}_n)$$

Recap: SVM – Dual Formulation

- **Maximize**

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m (\mathbf{x}_m^{\mathrm{T}}\mathbf{x}_n)$$

under the conditions

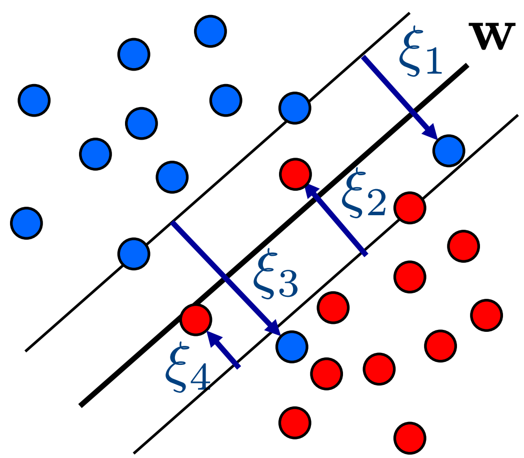
$$a_n \geq 0 \quad \forall n$$

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

- **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** $\mathcal{O}(N^2)$**.**

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.
  - $\xi_n = |t_n - y(\mathbf{x}_n)|$ for all other points.



Point on decision boundary: $\xi_n = 1$

Misclassified point: $\xi_n > 1$

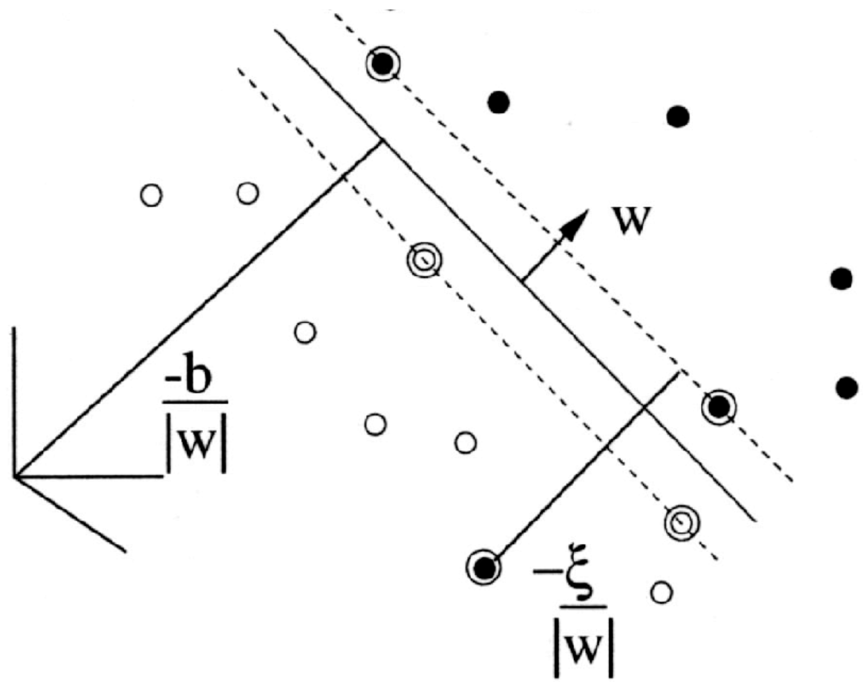
  - We do not have to set the slack variables ourselves!

$\Rightarrow$ They are jointly optimized together with $\mathbf{w}$.

Recap: SVM – Non-Separable Data

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

Trade-off parameter!



Recap: SVM – New Primal Formulation

- **New SVM Primal: Optimize**

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \underbrace{\sum_{n=1}^{N} a_n\left(t_n y(\mathbf{x}_n) - 1 + \xi_n\right)}_{\text{Constraint}} - \underbrace{\sum_{n=1}^{N}\mu_n\xi_n}_{\text{Constraint}}$$

Constraint

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n$$

Constraint

$$\xi_n \geq 0$$

- **KKT conditions**

$$\begin{aligned} a_n &\geq 0 \\ t_n y(\mathbf{x}_n) - 1 + \xi_n &\geq 0 \\ a_n\left(t_n y(\mathbf{x}_n) - 1 + \xi_n\right) &= 0 \end{aligned} \qquad \begin{aligned} \mu_n &\geq 0 \\ \xi_n &\geq 0 \\ \mu_n\xi_n &= 0 \end{aligned}$$

KKT:

$$\begin{aligned} \lambda &\geq 0 \\ f(\mathbf{x}) &\geq 0 \\ \lambda f(\mathbf{x}) &= 0 \end{aligned}$$

Recap: SVM – New Dual Formulation

- **New SVM Dual: Maximize**

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m (\mathbf{x}_m^{\mathrm{T}}\mathbf{x}_n)$$

under the conditions

$$0 \cdot a_n \cdot C$$

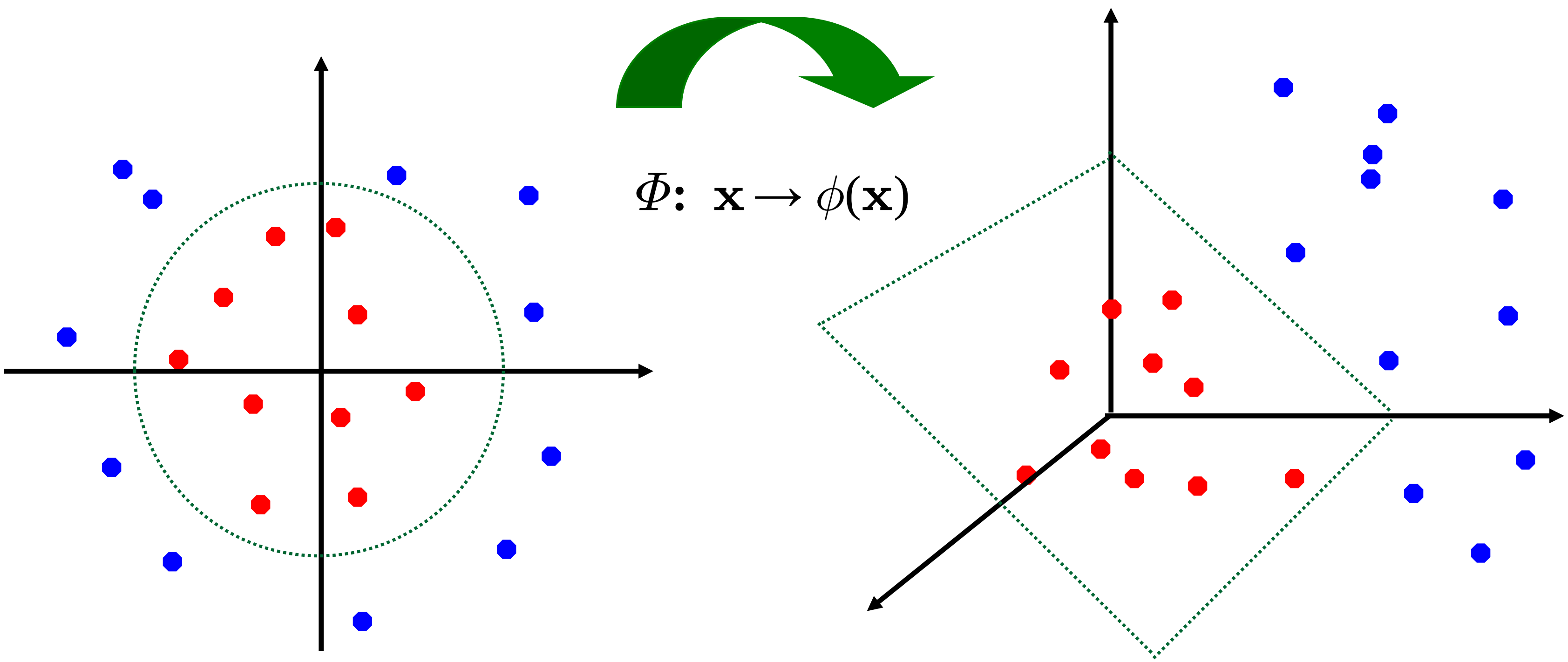
$$\sum_{n=1}^{N} a_n t_n = 0$$

This is all that changed!

- **This is again a quadratic programming problem**
  - $\Rightarrow$ Solve as before...

Recap: Nonlinear SVMs

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



$$\Phi\colon\ \mathbf{x} \rightarrow \phi(\mathbf{x})$$

Slide credit: Raymond Mooney

Recap: The Kernel Trick

- **Important observation**
  - $\phi(\mathbf{x})$ **only appears in the form of dot products** $\phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{y})$**:**

$$\begin{aligned} y(\mathbf{x}) &= \mathbf{w}^\mathrm{T}\phi(\mathbf{x}) + b \\ &= \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)^\mathrm{T}\phi(\mathbf{x}) + b \end{aligned}$$

  - **Define a so-called kernel function** $k(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{y})$**.**
  - **Now, in place of the dot product, use the kernel instead:**

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

  - **The kernel function *implicitly* maps the data to the higher-dimensional space (without having to compute** $\phi(\mathbf{x})$ **explicitly)!**

Recap: SVMs with Kernels

- Using kernels
  - Applying the kernel trick is easy. Just replace every dot product by a kernel function…

$$\mathbf{x}^{\mathrm{T}}\mathbf{y} \quad \rightarrow \quad k(\mathbf{x}, \mathbf{y})$$

  - …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 $\phi(\mathbf{x})$.
  - Kernel needs to fulfill Mercer's condition ($\rightarrow$ Lecture 4).

Recap: Nonlinear SVM – Dual Formulation

- **SVM Dual: Maximize**

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_m, \mathbf{x}_n)$$

under the conditions

$$0 \cdot a_n \cdot C$$

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

- **Classify new data points using**

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

RWTH AACHEN UNIVERSITY

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.
  - svmlight (http://svmlight.joachims.org/)
    - Command-line based interface
    - Source code available (in C)
    - Interfaces to Python, MATLAB, Perl, Java, DLL,…
  - libsvm (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)
    - 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, …

    $\Rightarrow$ 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

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

"Maximize the margin"

subject to the constraints

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n$$

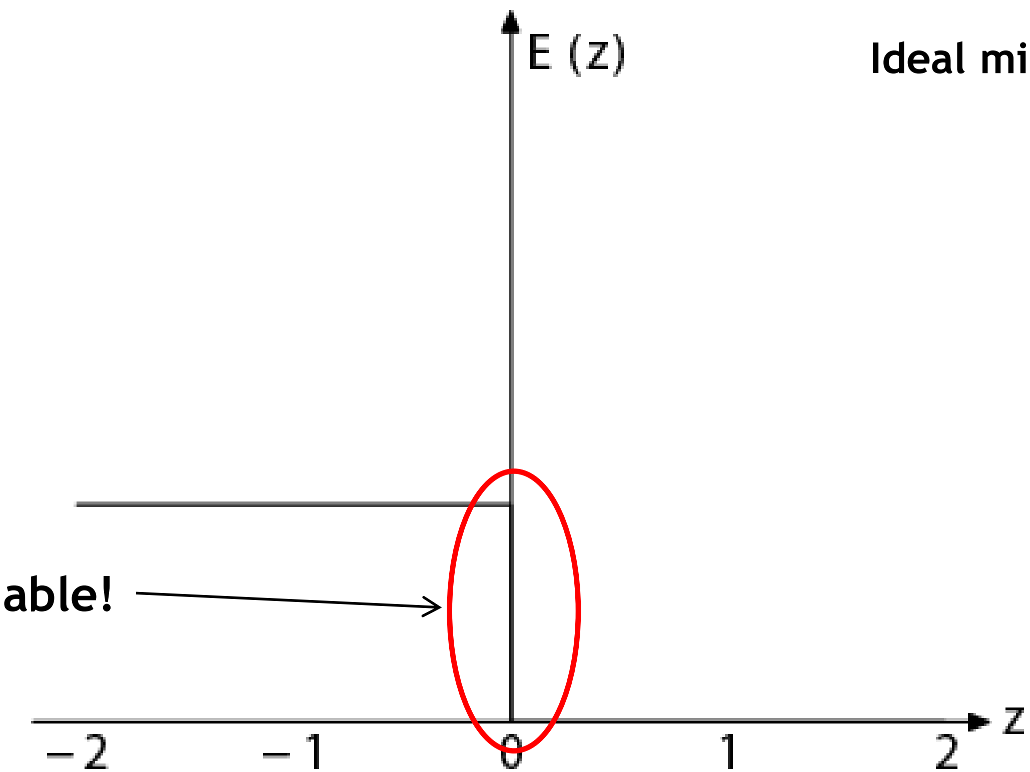
"Most points should be on the correct side of the margin"

- Different way of looking at it
  - We can reformulate the constraints into the objective function.

$$\min_{\mathbf{w}\in\mathbb{R}^D} \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{L_2 \text{ regularizer}} + \underbrace{C\sum_{n=1}^{N}[1 - t_n y(\mathbf{x}_n)]_+}_{\text{"Hinge loss"}}$$

where $[x]_+ := \max\{0,x\}$.

Error Functions (Loss Functions)



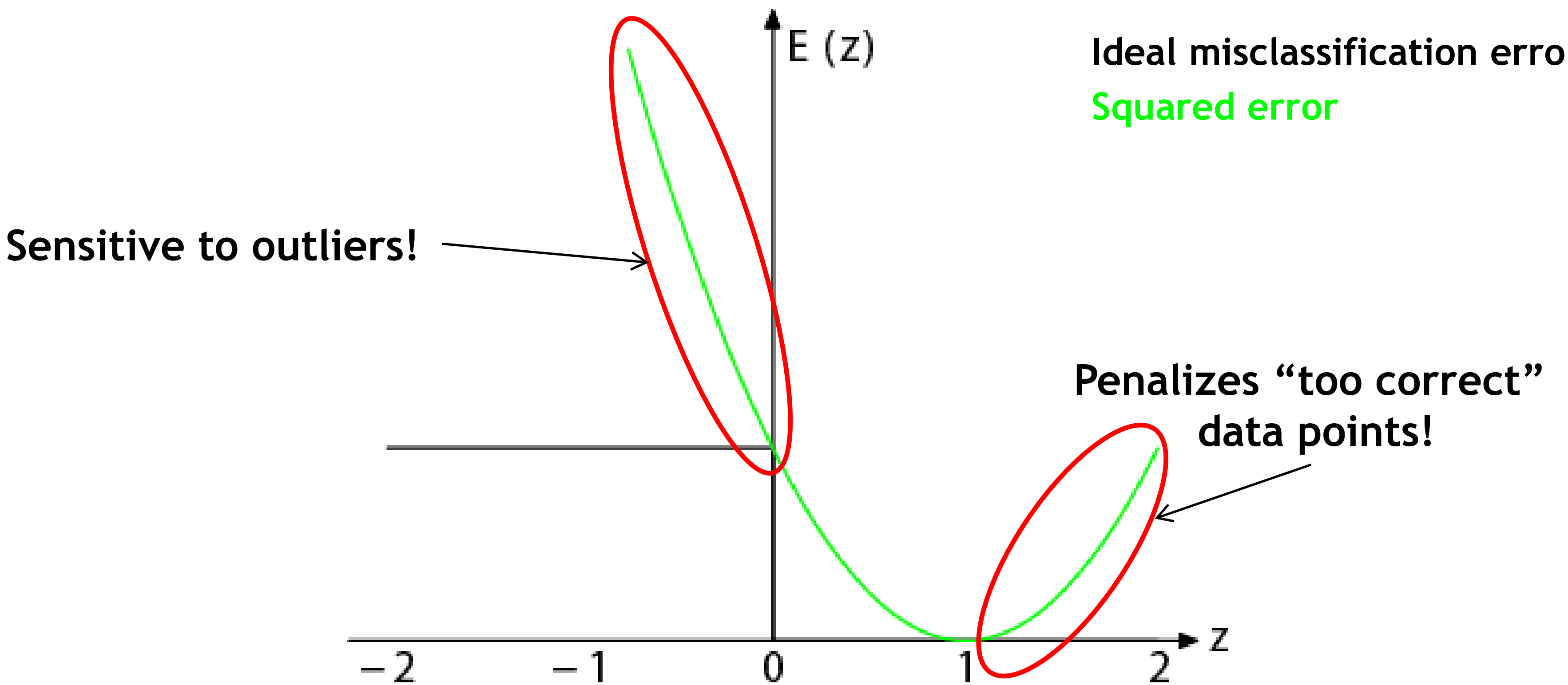
$z_n = t_n y(\mathbf{x}_n)$

- 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.

Image source: Bishop, 2006

Error Functions (Loss Functions)



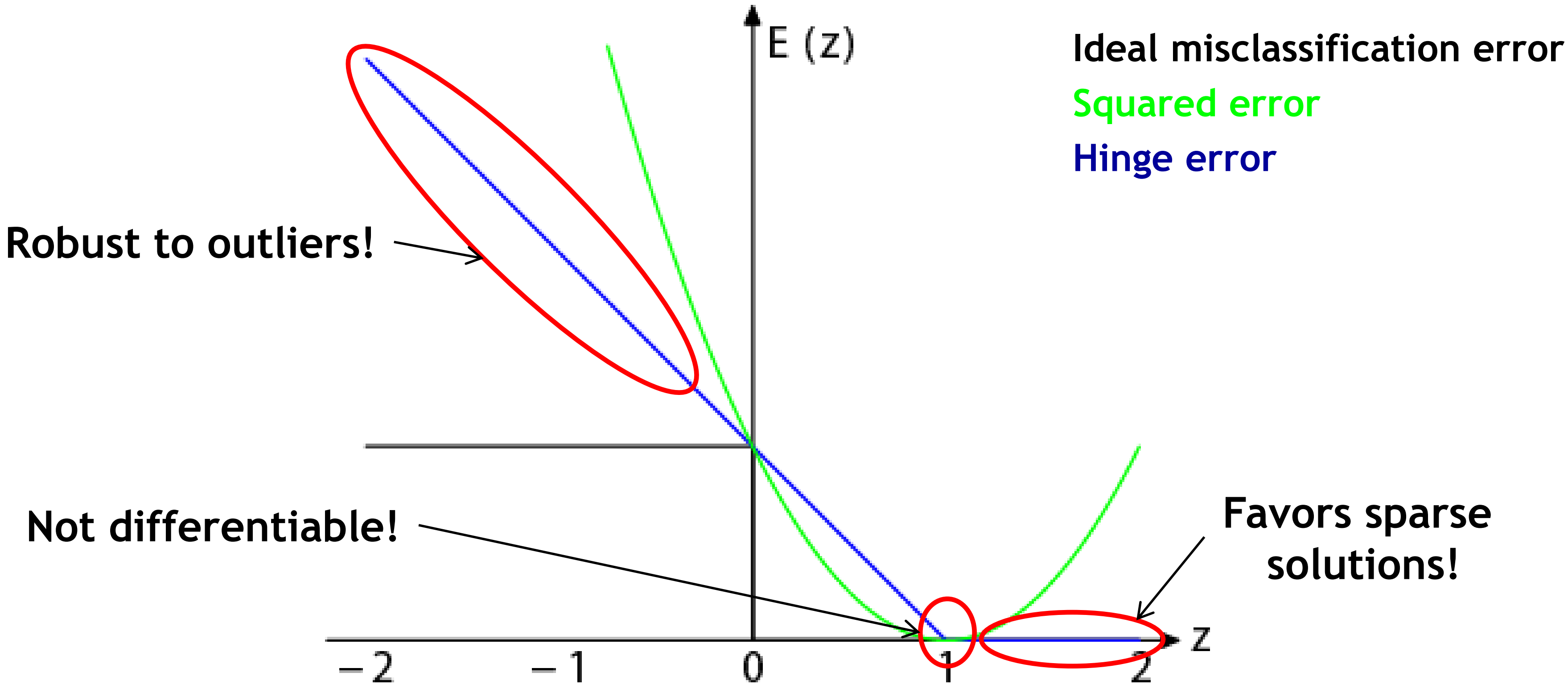
- Squared error used in Least-Squares Classification
  - Very popular, leads to closed-form solutions.
  - However, sensitive to outliers due to squared penalty.
  - Penalizes "too correct" data points

  $\Rightarrow$ Generally does not lead to good classifiers.

Image source: Bishop, 2006

RWTH AACHEN UNIVERSITY

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.

Image source: Bishop, 2006

SVM – Discussion

- SVM optimization function

$$\min_{\mathbf{w}\in\mathbb{R}^D} \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{\text{L}_2\text{ regularizer}} + \underbrace{C\sum_{n=1}^{N}[1 - t_n y(\mathbf{x}_n)]_+}_{\text{Hinge loss}}$$

- 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*

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

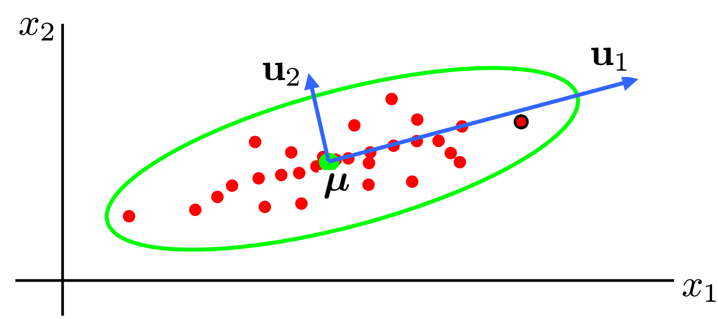
Topics of This Lecture

- 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 $\mathbf{x}_n \in \mathbb{R}^d$, PCA finds the directions of maximal covariance. Without loss of generality assume that $\sum_n \mathbf{x}_n = \mathbf{0}$.
  - The PCA directions $\mathbf{e}_1,\dots,\mathbf{e}_d$ are the eigenvectors of the covariance matrix

$$C = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T$$



sorted by their eigenvalue.

  - We can express $\mathbf{x}_n$ in PCA space by $F(\mathbf{x}_n) = \sum_{k=1}^{K} \langle \mathbf{x}_n, \mathbf{e}_k \rangle \mathbf{e}_k$
  - Lower-dim. coordinate mapping:

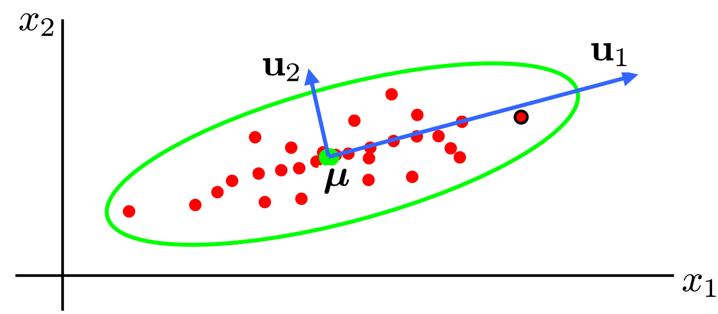
$$\mathbf{x}_n \mapsto \begin{pmatrix} \langle \mathbf{x}_n, \mathbf{e}_1 \rangle \\ \langle \mathbf{x}_n, \mathbf{e}_2 \rangle \\ \dots \\ \langle \mathbf{x}_n, \mathbf{e}_K \rangle \end{pmatrix} \in \mathbb{R}^K$$

Kernel-PCA

- **Kernel-PCA procedure**
  - Given samples $\mathbf{x}_n \in \mathcal{X}$, kernel $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with an implicit feature map ϕ: $\mathcal{X} \to \mathcal{H}$. Perform PCA in the Hilbert space $\mathcal{H}$.
  - The kernel-PCA directions $\mathbf{e}_1, \dots, \mathbf{e}_d$ are the eigenvectors of the covariance operator

$$C = \frac{1}{N} \sum_{n=1}^{N} \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T$$

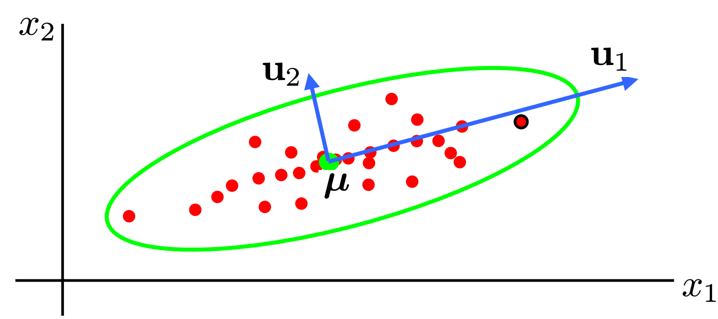
sorted by their eigenvalue.



  - Lower-dim. coordinate mapping: $\mathbf{x}_n \mapsto \begin{pmatrix} \langle \phi(\mathbf{x}_n), \mathbf{e}_1 \rangle \\ \langle \phi(\mathbf{x}_n), \mathbf{e}_2 \rangle \\ \dots \\ \langle \phi(\mathbf{x}_n), \mathbf{e}_K \rangle \end{pmatrix} \in \mathbb{R}^K$

Kernel-PCA

- **Kernel-PCA procedure**
  - Given samples $\mathbf{x}_n \in \mathcal{X}$, kernel $\mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ with an implicit feature map ϕ: $\mathcal{X} \rightarrow \mathcal{H}$. Perform PCA in the Hilbert space $\mathcal{H}$.
  - Equivalently, we can use the eigenvectors $\mathbf{e}'_k$ and eigenvalues λ_k of the kernel matrix

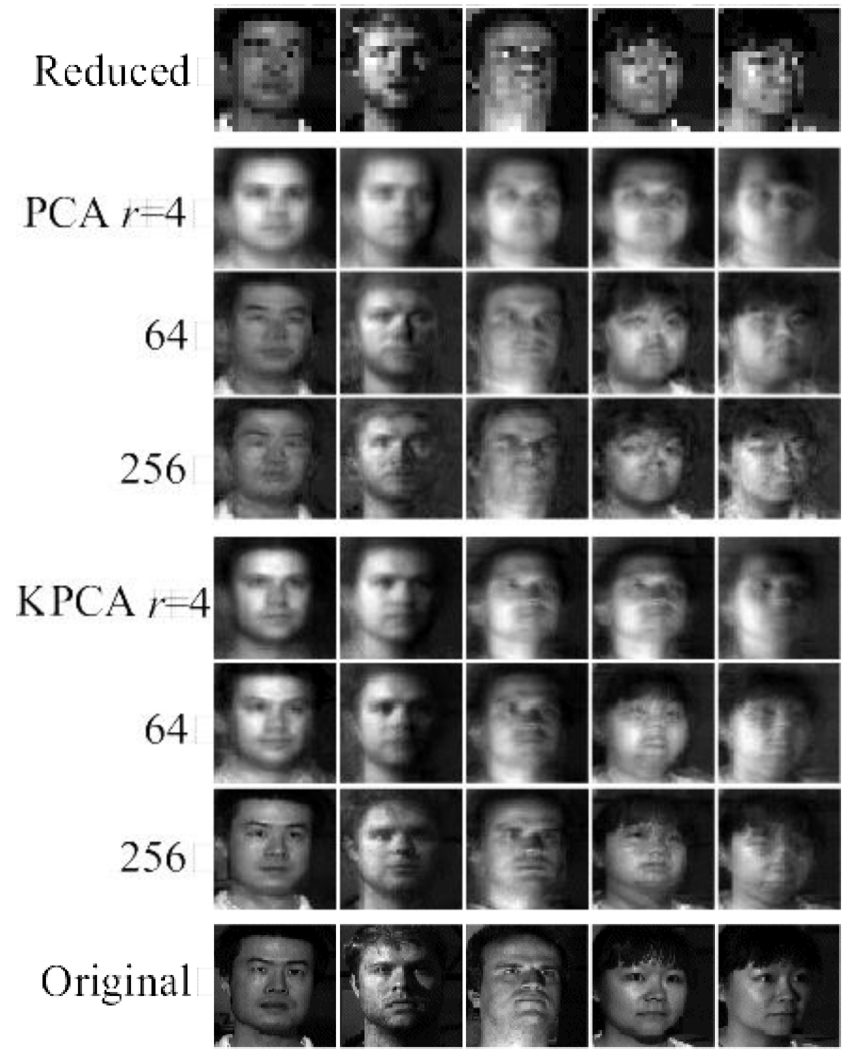


$$
\begin{aligned}
K &= (\langle \phi(\mathbf{x}_m), \phi(\mathbf{x}_n) \rangle)_{m,n=1,\ldots,N} \\
&= (k(\mathbf{x}_m, \mathbf{x}_n))_{m,n=1,\ldots,N}
\end{aligned}
$$

  - Coordinate mapping: $\mathbf{x}_n \mapsto (\sqrt{\lambda_1}\mathbf{e}_1', \ldots, \sqrt{\lambda_K}\mathbf{e}_K')$

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



Reconstruction in r dimensions

Kim, Jung, Kim, [Face Recognition using Kernel Principal Component Analysis](), Signal Processing Letters, 2002.

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 $\mathcal{X} \to \mathbb{R}^{d'}$, so it provides a vectorial representation of non-vectorial data.

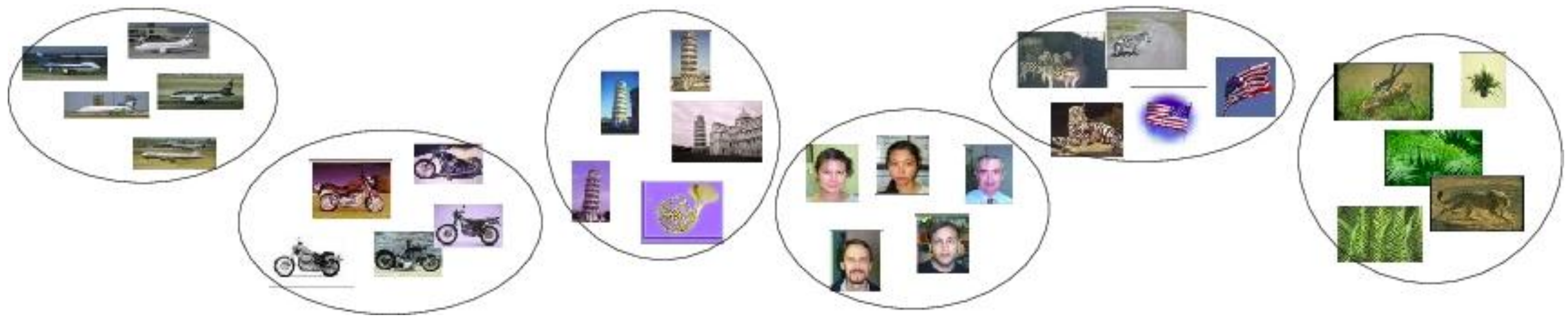
  $\Rightarrow$ We can apply algorithms that only work in vector spaces to data that is not in a vector representation.

- Example: k-Means clustering
  - Given $\mathbf{x}_1,\ldots,\mathbf{x}_n \in \mathcal{X}$.
  - Choose a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.
  - Apply kernel-PCA to obtain vectorial $\mathbf{v}_1,\ldots,\mathbf{v}_n \in \mathbb{R}^{d'}$.
  - Cluster $\mathbf{v}_1,\ldots,\mathbf{v}_n \in \mathbb{R}^{d'}$ using *K*-Means.

  $\Rightarrow$ $\mathbf{x}_1,\ldots,\mathbf{x}_n$ are clustered based on the similarity defined by k.

Slide credit: Christoph Lampert

Example: Unsupervised Object Categorization

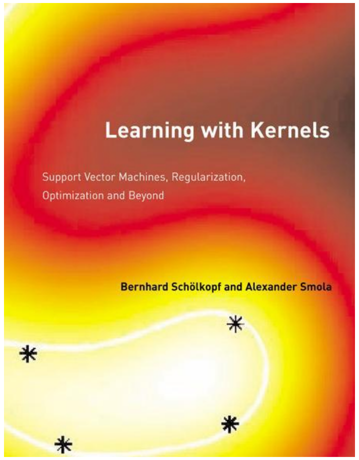


- Automatically group images that show similar objects
  - Represent images by bag-of-word histograms
  - Perform Kernel k-Means Clustering
  - $\Rightarrow$ Observation: Clusters get better if we use a good image kernel (e.g., χ^2) instead of plain k-Means (linear kernel).

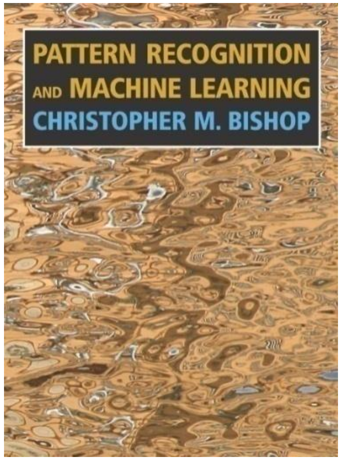
T. Tuytelaars, C. Lampert, M. Blaschko, W. Buntine, Unsupervised object discovery: a comparison, IJCV, 2009.]

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/