Semi-supervised Learning with Kernels

Scientific Advisor: **Zoltán Kása**
Ph.D Student: **Zalán Bodó**

Babeș–Bolyai University, Cluj-Napoca
Faculty of Mathematics and Computer Science

Outline

Kernel methods

- James Mercer (1909): any continuous symmetric, positive semi-definite kernel function can be expressed as a dot product in a high-dimensional space
- first application: M. Aizerman, E. Braverman, and L. Rozonoer, 1964
- famous application: Boser, Guyon, and Vapnik, 1992 (SVM)
- linear algorithms $\rightarrow$ non-linear algorithms
- feature mapping: $\phi : X \rightarrow \mathcal{H}$
- kernels: $k(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle = \phi(\boldsymbol{x})' \phi(\boldsymbol{z})$
- covers all geometric constructions that can be formulated in terms of angles, lengths and distances

Kernel trick

Given an algorithm which is formulated in terms of a positive definite kernel $k(\cdot, \cdot)$, one can construct an alternative algorithm by replacing $k(\cdot, \cdot)$ by another positive definite kernel $\widetilde{k}(\cdot, \cdot)$.

Kernel methods

- James Mercer (1909): any continuous symmetric, positive semi-definite kernel function can be expressed as a dot product in a high-dimensional space
- first application: M. Aizerman, E. Braverman, and L. Rozonoer, 1964
- famous application: Boser, Guyon, and Vapnik, 1992 (SVM)
- linear algorithms $\rightarrow$ non-linear algorithms
- feature mapping: $\phi : X \rightarrow \mathcal{H}$
- kernels: $k(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle = \phi(\boldsymbol{x})' \phi(\boldsymbol{z})$
- covers all geometric constructions that can be formulated in terms of angles, lengths and distances

Kernel trick

Given an algorithm which is formulated in terms of a positive definite kernel $k(\cdot, \cdot)$, one can construct an alternative algorithm by replacing $k(\cdot, \cdot)$ by another positive definite kernel $\widetilde{k}(\cdot, \cdot)$.

Examples of general purpose kernels

- linear:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x}'\boldsymbol{z}$$

- polynomial:

$$k(\boldsymbol{x}, \boldsymbol{z}) = (a\boldsymbol{x}'\boldsymbol{z} + b)^c$$

- Gaussian:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|^2}{2\sigma^2}\right)$$

- sigmoid:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \tanh(a\boldsymbol{x}'\boldsymbol{z} + r)$$

Examples of general purpose kernels

- linear:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x}'\boldsymbol{z}$$

- polynomial:

$$k(\boldsymbol{x}, \boldsymbol{z}) = (a\boldsymbol{x}'\boldsymbol{z} + b)^c$$

- Gaussian:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|^2}{2\sigma^2}\right)$$

- sigmoid:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \tanh(a\boldsymbol{x}'\boldsymbol{z} + r)$$

Examples of general purpose kernels

- linear:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x}'\boldsymbol{z}$$

- polynomial:

$$k(\boldsymbol{x}, \boldsymbol{z}) = (a\boldsymbol{x}'\boldsymbol{z} + b)^c$$

- Gaussian:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|^2}{2\sigma^2}\right)$$

- sigmoid:

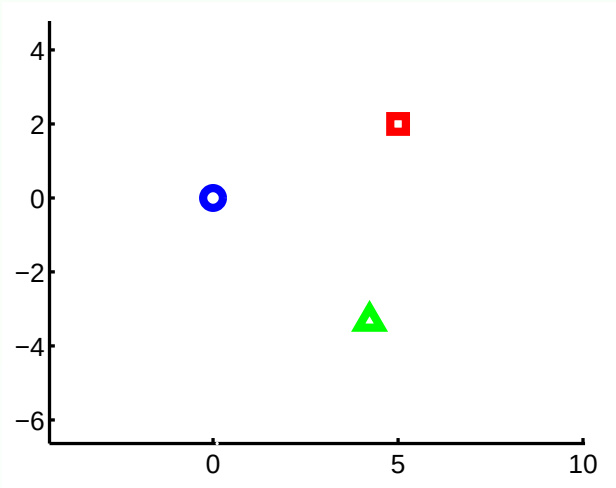
$$k(\boldsymbol{x}, \boldsymbol{z}) = \tanh(a\boldsymbol{x}'\boldsymbol{z} + r)$$

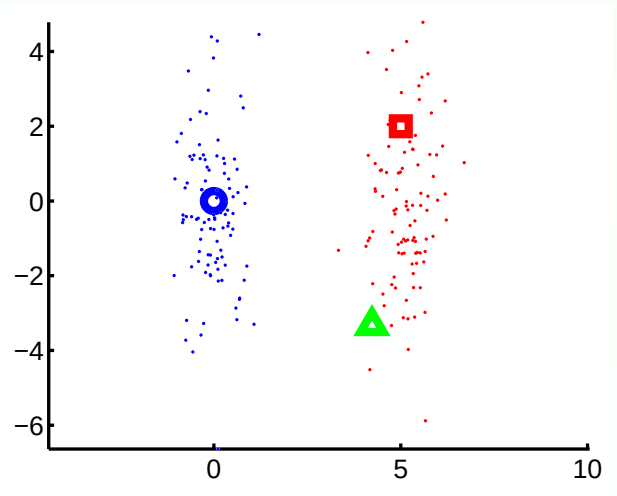
Examples of general purpose kernels

- linear:
$$k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x}'\boldsymbol{z}$$
- polynomial:
$$k(\boldsymbol{x}, \boldsymbol{z}) = (a\boldsymbol{x}'\boldsymbol{z} + b)^c$$
- Gaussian:
$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|^2}{2\sigma^2}\right)$$
- sigmoid:
$$k(\boldsymbol{x}, \boldsymbol{z}) = \tanh(a\boldsymbol{x}'\boldsymbol{z} + r)$$

Semi-supervised learning (SSL)

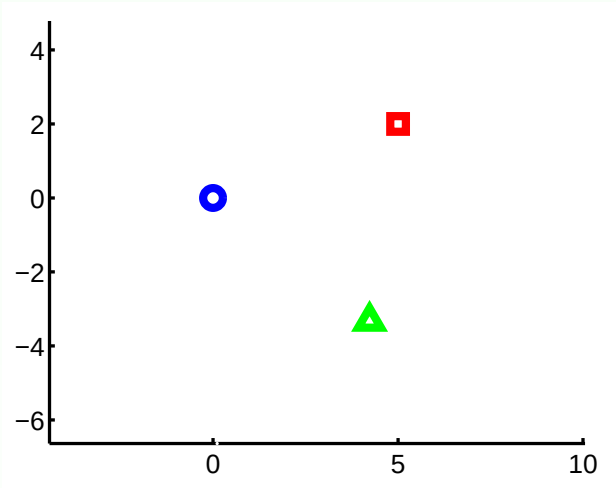
- supervised learning:
  $D = \{(\boldsymbol{x}_i, y_i) \,|\, \boldsymbol{x}_i \in X \subseteq \mathbb{R}^d, y_i \in \{-1, +1\}, i = 1, \ldots, \ell\}$;
  find $f : X \to \{-1, +1\}$ which agrees with D
- semi-supervised learning:
  $D = \{(\boldsymbol{x}_i, y_i) \,|\, i = 1, \ldots, \ell\} \cup \{\boldsymbol{x}_j \,|\, j = 1, \ldots, u\}$, $\ell \ll u$,
  $N = \ell + u$;
  - **inductive**: find $f : X \to \{-1, +1\}$ which agrees with D + use the information of D_U
  - **transductive**: find $f : D_U \to \{-1, +1\}$ by using $D = D_L \cup D_U$

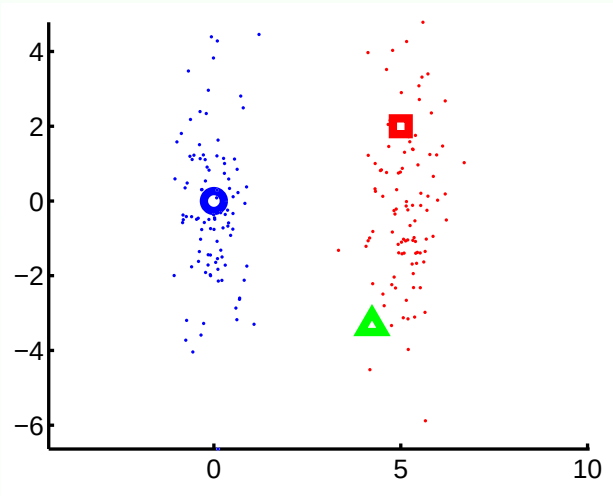




Semi-supervised learning (SSL)

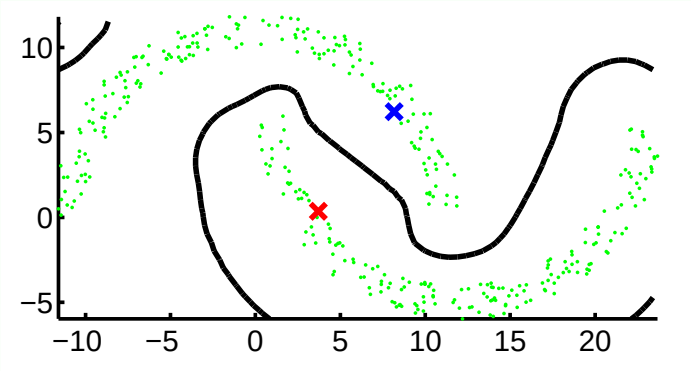
- supervised learning:
  $D = \{(\boldsymbol{x}_i, y_i) \,|\, \boldsymbol{x}_i \in X \subseteq \mathbb{R}^d, y_i \in \{-1, +1\}, i = 1, \ldots, \ell\}$;
  find $f : X \to \{-1, +1\}$ which agrees with D
- semi-supervised learning:
  $D = \{(\boldsymbol{x}_i, y_i) \,|\, i = 1, \ldots, \ell\} \cup \{\boldsymbol{x}_j \,|\, j = 1, \ldots, u\}$, $\ell \ll u$,
  $N = \ell + u$;
  - **inductive**: find $f : X \to \{-1, +1\}$ which agrees with D + use the information of D_U
  - **transductive**: find $f : D_U \to \{-1, +1\}$ by using $D = D_L \cup D_U$





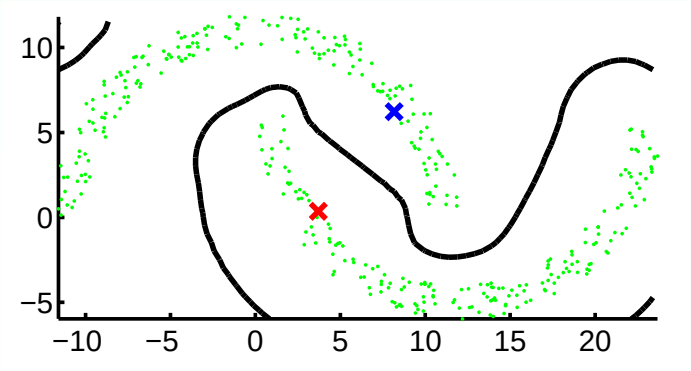
Assumptions in SSL

- **smoothness assumption**: *If two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in a high density region are close, then so should be the corresponding outputs y_i and y_j.*
- **cluster assumption**: *If two points are in the same cluster, they are likely to be of the same class.*
- **manifold assumption**: *The high dimensional data lie roughly on a low dimensional manifold.* – regarding dimensionality; but if manifold = approximation of the high-dimensional region $\Rightarrow$ smoothness assumption



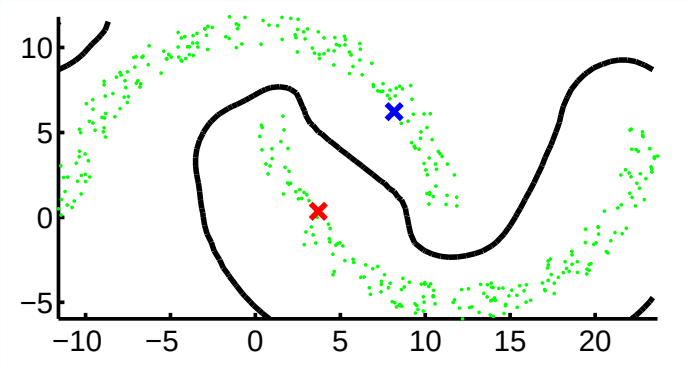
Assumptions in SSL

- **smoothness assumption**: *If two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in a high density region are close, then so should be the corresponding outputs y_i and y_j.*
- **cluster assumption**: *If two points are in the same cluster, they are likely to be of the same class.*
- **manifold assumption**: *The high dimensional data lie roughly on a low dimensional manifold.* – regarding dimensionality; but if manifold = approximation of the high-dimensional region $\Rightarrow$ smoothness assumption



Assumptions in SSL

- **smoothness assumption**: *If two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in a high density region are close, then so should be the corresponding outputs y_i and y_j.*
- **cluster assumption**: *If two points are in the same cluster, they are likely to be of the same class.*
- **manifold assumption**: *The high dimensional data lie roughly on a low dimensional manifold.* – regarding dimensionality; but if manifold = approximation of the high-dimensional region $\Rightarrow$ smoothness assumption



Classes of SSL

- Generative models
- Low-density separation
- Graph-based methods
- **Change of representation**

Classes of SSL

- Generative models
- Low-density separation
- Graph-based methods
- **Change of representation**

- model class conditional density $P(\boldsymbol{x}|y)$ and class priors $P(y)$ and use the Bayes theorem for calculating posteriors, which are used for classification
- **e.g.** Naive Bayes + EM
  1. Train the classifier on the training examples; determine probabilities $P(\boldsymbol{d}_i \mid c_j)$ and $P(c_j)$.
  2. Repeat while there is improvement:

     E-step: Classify unlabeled examples using the trained classifier.
     M-step: Re-estimate the classifier using the previously classified examples; recalculate $P(\boldsymbol{d}_i \mid c_j)$ and $P(c_j)$.

Classes of SSL

- Generative models
- Low-density separation
- Graph-based methods
- **Change of representation**

- push away a decision boundary from labeled and unlabeled data; natural choice: use a large-margin classifier
- **e.g.** Transductive SVM (TSVM)

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\
\text{s.t.} \quad & y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \geq 1, \quad i = 1, \ldots, \ell \\
& y_j(\boldsymbol{w}'\boldsymbol{x}_j + b) \geq 1, \quad j = \ell + 1, \ldots, N \\
& y_j \in \{-1, +1\}, \quad j = \ell + 1, \ldots, N
\end{aligned}
$$

Classes of SSL

- Generative models
- Low-density separation
- Graph-based methods
- **Change of representation**

- use the graph of the labeled and unlabeled data, represented by weight matrix $\boldsymbol{W}$; use graph Laplacian $\boldsymbol{L} - \boldsymbol{W}$
- **e.g.** Label Propagation
  1. Compute $\boldsymbol{Y}(t+1) = \boldsymbol{P}\,\boldsymbol{Y}(t)$.
  2. Reset the labeled data, $\boldsymbol{Y}_L(t+1) = \boldsymbol{Y}_L(0)$.
  3. $t = t+1$ and repeat the above steps until convergence.

Classes of SSL

- Generative models
- Low-density separation
- Graph-based methods
- **Change of representation**

- find some structure in the whole data set; use the information provided by the labeled and unlabeled data set to create a new representation
  1. Build the new representation – new distance, dot-product or kernel – of the learning examples.
  2. Use a supervised learning method for obtaining the decision function employing the new representation obtained in the previous step.
- **e.g.** PCA, KPCA, LLE, ISOMAP, etc.

Data-dependent kernels

- supervised learning + data-dependent kernels = **semi-supervised learning**
- conventional kernels: given data sets $D_1 \neq D_2$, $\boldsymbol{x}, \boldsymbol{z} \in D_1 \cap D_2$

$$k(\boldsymbol{x}, \boldsymbol{z}) = k(\boldsymbol{x}, \boldsymbol{z})$$

- data-dependent kernels: given data sets $D_1 \neq D_2$, $\boldsymbol{x}, \boldsymbol{z} \in D_1 \cap D_2$

$$k(\boldsymbol{x}, \boldsymbol{z}; D_1) \eqsim k(\boldsymbol{x}, \boldsymbol{z}; D_2)$$

"$\eqsim$" reads as "not necessarily equal"

Data-dependent kernels

- supervised learning + data-dependent kernels = **semi-supervised learning**
- conventional kernels: given data sets $D_1 \neq D_2$, $\boldsymbol{x}, \boldsymbol{z} \in D_1 \cap D_2$

$$k(\boldsymbol{x}, \boldsymbol{z}) = k(\boldsymbol{x}, \boldsymbol{z})$$

- data-dependent kernels: given data sets $D_1 \neq D_2$, $\boldsymbol{x}, \boldsymbol{z} \in D_1 \cap D_2$

$$k(\boldsymbol{x}, \boldsymbol{z}; D_1) \eqsim k(\boldsymbol{x}, \boldsymbol{z}; D_2)$$

"$\eqsim$" reads as "not necessarily equal"

Kernels proposed

- Wikipedia-based kernels for text categorization
  - kernel construction for textual data based on Wikipedia
  - inclusion of the link structure of Wikipedia into the kernel
  - concept weighting in the kernel using the PageRank algorithm
- hierarchical cluster kernels
  - proposal of a general framework for constructing hierarchical cluster kernels
  - proposal of graph-based hierarchical cluster kernels
- reweighting cluster kernels
  - construction of cluster kernels using the Hadamard product property of positive semi-definite kernels
  - introduction of the Gaussian reweighting kernel
  - introduction of two reweighting kernels using the dot products of cluster membership vectors

Kernels proposed

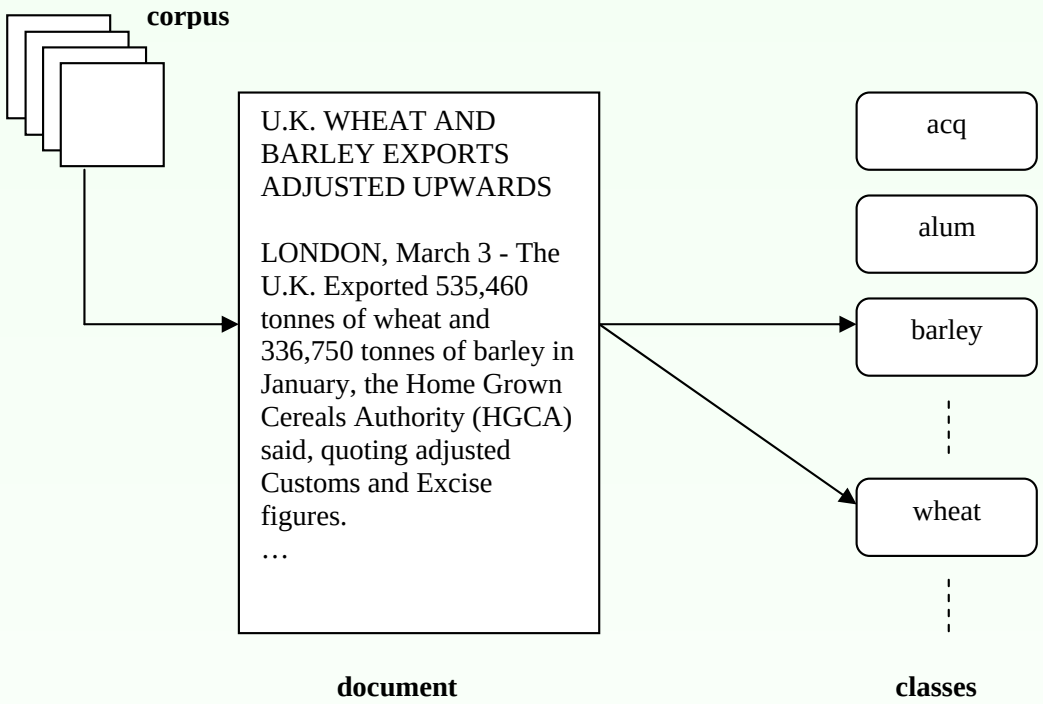
- Wikipedia-based kernels for text categorization
  - kernel construction for textual data based on Wikipedia
  - inclusion of the link structure of Wikipedia into the kernel
  - concept weighting in the kernel using the PageRank algorithm
- hierarchical cluster kernels
  - proposal of a general framework for constructing hierarchical cluster kernels
  - proposal of graph-based hierarchical cluster kernels
- reweighting cluster kernels
  - construction of cluster kernels using the Hadamard product property of positive semi-definite kernels
  - introduction of the Gaussian reweighting kernel
  - introduction of two reweighting kernels using the dot products of cluster membership vectors

Kernels proposed

- Wikipedia-based kernels for text categorization
  - kernel construction for textual data based on Wikipedia
  - inclusion of the link structure of Wikipedia into the kernel
  - concept weighting in the kernel using the PageRank algorithm
- hierarchical cluster kernels
  - proposal of a general framework for constructing hierarchical cluster kernels
  - proposal of graph-based hierarchical cluster kernels
- reweighting cluster kernels
  - construction of cluster kernels using the Hadamard product property of positive semi-definite kernels
  - introduction of the Gaussian reweighting kernel
  - introduction of two reweighting kernels using the dot products of cluster membership vectors

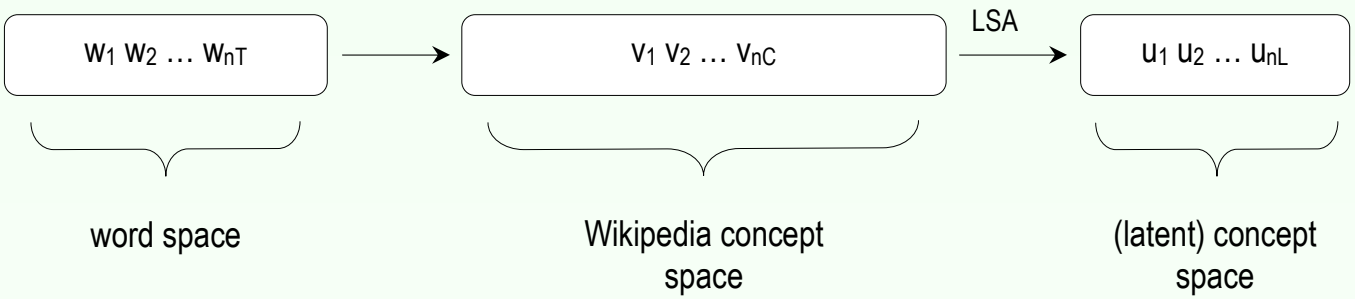
Wikipedia-based kernels for text categorization

Text categorization



- TC = representation + feature selection + learning
- representation: Vector Space Model
- feature selection: wrappers, embedded methods, filters
- learning: kernel methods $\Rightarrow$ finding a good text similarity metric (kernel)

The proposed kernel



- transform documents to Wikipedia concept space
- use LSA for filtering out irrelevant concepts
- (+ link structure of Wikipedia, concept weighting with PageRank)
- Wikipedia matrix:

$$\underbrace{\begin{pmatrix} c_{11} & c_{12} & \dots & c_{1n_T} \\ c_{21} & c_{22} & \dots & c_{2n_T} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n_C1} & c_{n_C2} & \dots & c_{n_Cn_T} \end{pmatrix}}_{W} \cdot \underbrace{\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{n_T} \end{pmatrix}}_{d}$$

- Wikipedia (November, 2006) – 1.6×10^6 articles $\approx$ 8 Gb of textual data
- Wikipedia kernel: $k(\boldsymbol{d}_i, \boldsymbol{d}_j) = \boldsymbol{d}_i' \boldsymbol{W}' \boldsymbol{W} \boldsymbol{d}_j$ [Gabrilovich & Markovitch, 2007]
- dimensionality reduction: use LSA/PCA/SVD

$$\boldsymbol{W} \approx \widehat{\boldsymbol{W}} = \boldsymbol{U} \boldsymbol{S}_k \boldsymbol{V}'$$

the new Wikipedia kernel becomes

$$k(\boldsymbol{d}_i, \boldsymbol{d}_j) = \boldsymbol{d}_i' \widehat{\boldsymbol{W}}' \widehat{\boldsymbol{W}} \boldsymbol{d}_j = \boldsymbol{d}_i' \boldsymbol{V}_k \boldsymbol{S}_k^2 \boldsymbol{V}_k' \boldsymbol{d}_j$$

- replacing $\boldsymbol{S}_k$ by $\boldsymbol{I}_k$ (too large weights):

$$k(\boldsymbol{d}_i, \boldsymbol{d}_j) = \boldsymbol{d}_i' \boldsymbol{V}_k \boldsymbol{V}_k' \boldsymbol{d}_j$$

- efficient computation of $\boldsymbol{V}_k$: eigendecomposition of $\boldsymbol{W}'\boldsymbol{W}$ (of size $n_T \times n_T$)

$$\boldsymbol{W}'\boldsymbol{W} = \boldsymbol{V} \boldsymbol{S}^2 \boldsymbol{V}'$$

Link structure of Wikipedia

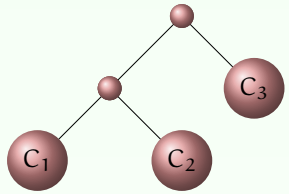
- $\boldsymbol{E}$ – link matrix, $E_{ij} = 1$ if $i \to j$
- use $\widetilde{\boldsymbol{W}} = \boldsymbol{E}'\boldsymbol{W}$ in the kernel

Concept weighting

- weighting Wikipedia concepts using PageRank
- use $\widetilde{\boldsymbol{W}} = \mathrm{diag}(\widetilde{\boldsymbol{r}})\boldsymbol{W}$ in the kernel

Hierarchical cluster kernels

- use distances obtained by agglomerative clustering (single, complete & average linkage)
- nice property: yields *ultrametric* matrices
- ultrametric property: one merges three clusters:



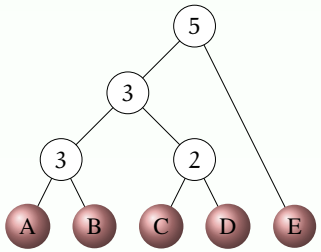
then: if

$$\begin{aligned} D(C_1, C_2) &\leq D(C_1, C_3) \\ D(C_1, C_2) &\leq D(C_2, C_3) \end{aligned}$$

then

$$D(C_1, C_2) \leq D(C_{12}, C_3)$$

- an ultrametric tree:



Theorem (Fischer et al., 2003)

Given an ultrametric $\boldsymbol{M}$*, the matrix* $-\frac{1}{2}\boldsymbol{M}^c = -\frac{1}{2}\boldsymbol{JMJ}$ *is a Gram matrix containing the dot products of the vectors* $\boldsymbol{z}_i$*,* $i = 1, 2, \ldots, N$*, whose squared Euclidean distances are contained in matrix* $\boldsymbol{M}$*.*

2 kernel construction proposals:

- HCK (Hierarchical Cluster Kernel) – using the cluster assumption
- gHCK (Graph-based HCK) – using the cluster assumption + manifold assumption; uses graph distances (similarly to ISOMAP)

HCK/gHCK algorithm:

-2. *Determine the k nearest neighbors or an ϵ-neighborhood of each point and take all the distances to other points equal to zero.*

-1. *Compute shortest paths for every pair of points – using for example Dijkstra's algorithm.*

0. *Use these distances in clustering for the pointwise distance $d(\cdot, \cdot)$ when calculating the linkage distance.*

1. Perform an agglomerative clustering on the labeled and unlabeled data – use one of the linkage functions of single, complete or average linkage clusterings.
2. Define matrix $\boldsymbol{M}$ with entries M_{ij} = linkage distance in the resulting ultrametric tree at the lowest common subsumer of i and j; $M_{ii} = 0$, $\forall i$.
3. Define the kernel matrix as $\boldsymbol{K} = -\frac{1}{2}\boldsymbol{JMJ}$.

HCK/gHCK algorithm:

-2. *Determine the k nearest neighbors or an ϵ-neighborhood of each point and take all the distances to other points equal to zero.*

-1. *Compute shortest paths for every pair of points – using for example Dijkstra's algorithm.*

0. *Use these distances in clustering for the pointwise distance $d(\cdot,\cdot)$ when calculating the linkage distance.*

1. Perform an agglomerative clustering on the labeled and unlabeled data – use one of the linkage functions of single, complete or average linkage clusterings.

2. Define matrix $\boldsymbol{M}$ with entries M_{ij} = linkage distance in the resulting ultrametric tree at the lowest common subsumer of i and j; $M_{ii} = 0$, $\forall i$.

3. Define the kernel matrix as $\boldsymbol{K} = -\frac{1}{2}\boldsymbol{JMJ}$.

Reweighting cluster kernels

- idea borrowed from bagged cluster kernel [Weston et al., 2004]
- reweighting conventional kernels according to some clustering of the data
- kernel combinations: $\boldsymbol{K}_1 + \boldsymbol{K}_2$, $a \cdot \boldsymbol{K}$, $\boldsymbol{K}_1 \odot \boldsymbol{K}_2$
- cluster kernel: $\boldsymbol{K} = \boldsymbol{K}_{\mathrm{rw}} \odot \boldsymbol{K}_{\mathrm{b}}$ where
  - $\boldsymbol{K}_{\mathrm{b}}$ = base kernel (e.g. Gaussian, polynomial, etc.)
  - $\boldsymbol{K}_{\mathrm{rw}}$ = reweighting kernel
  - $\boldsymbol{K}$ = resulting cluster kernel used in the learning algorithm

3 kernel construction proposals:

- Gaussian reweighting kernel:

$$k_{\mathrm{rw}}(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{U}_{\cdot\boldsymbol{x}} - \boldsymbol{U}_{\cdot\boldsymbol{z}}\|^2}{2\sigma^2}\right)$$

- dot product-based reweighting kernels:

$$\boldsymbol{K}_{\mathrm{rw}} = \boldsymbol{U}'\boldsymbol{U} + \alpha \cdot \mathbf{1}\mathbf{1}', \quad \alpha \in [0, 1)$$

$$\boldsymbol{K}_{\mathrm{rw}} = \beta \cdot \boldsymbol{U}'\boldsymbol{U} + \mathbf{1}\mathbf{1}', \quad \beta \in (0, \infty)$$

where $\boldsymbol{U}$ of size $K \times N$ matrix of cluster membership vectors (columns)

3 kernel construction proposals:

- Gaussian reweighting kernel:

$$k_{\mathrm{rw}}(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{U}_{\cdot\boldsymbol{x}} - \boldsymbol{U}_{\cdot\boldsymbol{z}}\|^2}{2\sigma^2}\right)$$

- dot product-based reweighting kernels:

$$\boldsymbol{K}_{\mathrm{rw}} = \boldsymbol{U}'\boldsymbol{U} + \alpha \cdot \mathbf{1}\mathbf{1}', \quad \alpha \in [0, 1)$$

$$\boldsymbol{K}_{\mathrm{rw}} = \beta \cdot \boldsymbol{U}'\boldsymbol{U} + \mathbf{1}\mathbf{1}', \quad \beta \in (0, \infty)$$

where $\boldsymbol{U}$ of size $K \times N$ matrix of cluster membership vectors (columns)

Experiments

Other data-dependent kernels

- linear & Gaussian kernels (not data-dependent)
- ISOMAP
- neighborhood kernel
- bagged cluster kernel
- Laplacian SVMs
- multi-type cluster kernel

Evaluation data sets

Data set	Classes	Dimension	Points	Comment
Reuters-21578	90	5209	9603/3299	sparse discrete; for Wikipedia-based kernels only
USPS	2	241	1500	imbalanced
Digit1	2	241	1500	artificial
COIL2	2	241	1500	
Text	2	11 960	1500	sparse discrete

Wikipedia-based kernel

	#eigv	mP	mR	mBEP	mF1	MP	MR	MBEP	MF1
$\chi^2$5209	–	88.46	84.59	86.52	86.48	71.61	61.25	66.43	66.02
$\chi^2$4601	–	87.88	83.09	85.49	85.42	64.21	54.62	59.41	59.03
Wikipedia covariance	–	48.95	35.42	42.18	41.10	8.79	5.35	7.07	6.65
LSA	4500	88.05	83.65	85.85	85.80	62.48	54.38	58.43	58.15
LSA+links	4500	87.89	83.71	85.80	85.75	65.11	56.05	60.58	60.24
LSA+PageRank	4000	87.44	83.68	85.56	85.52	59.75	53.86	56.80	56.65

Table: Performance on the Reuters corpus in percentage. Notation: mP=micro-precision, mR=micro-recall, mBEP=micro-breakeven, mF1=micro-F_1, MP=macro-precision, MR=macro-recall, MBEP=macro-breakeven, MF1=macro-F_1

– best result (baseline $\chi^2$5209),

– best result (baseline $\chi^2$4601)

Hierarchical and reweighting cluster kernels

	USPS		Digit1		COIL2		Text	
	10	100	10	100	10	100	10	100
linear	72.82	86.43	81.07	90.86	60.74	80.43	58.26	67.86
Gaussian	80.07	89.71	56.11	93.86	57.38	82.50	59.06	56.43
ISOMAP	85.10	86.71	94.43	97.43	62.62	80.64	59.80	72.43
neighborhood	76.31	94.14	87.11	94.21	64.43	84.43	51.68	62.79
bagged	87.38	92.79	93.29	96.93	71.28	85.57	63.29	66.14
multi-type, step	80.07	92.86	91.01	91.29	55.77	84.86	53.56	74.79
multi-type, linear step	80.07	92.86	91.01	91.36	55.77	84.86	53.02	75.29
multi-type, polynomial	80.07	80.29	48.86	65.07	54.23	82.29	50.60	56.71
LapSVM, Gaussian	81.95	95.93	84.50	97.64	76.64	97.71	63.42	62.50

Table: Baseline results: Accuracy results using different kernels. The results are given in percentage. For each data set the best results are put in a framebox.

– best, – second best, – third best results

	USPS		Digit1		COIL2		Text	
	10	100	10	100	10	100	10	100
HCK, single	80.07	81.79	48.86	70.21	67.85	96.00	66.78	73.14
HCK, complete	82.01	89.50	60.67	89.71	55.64	86.36	50.27	49.57
HCK, average	81.48	92.86	71.75	93.79	68.05	91.71	64.63	50.14
gHCK, single	80.07	81.79	48.86	70.21	60.60	93.86	66.78	73.14
gHCK, complete	88.26	95.64	75.50	93.71	68.52	88.79	56.17	67.71
gHCK, average	89.26	95.64	94.70	95.21	60.54	90.64	47.32	66.86
RCK1, k-means	84.45	92.98	83.14	94.28	58.55	83.76	–	–
RCK1, hierarchical	86.17	95.29	89.06	94.94	62.08	85.93	62.35	68.07
RCK1, spectral	81.43	90.87	88.32	95.20	58.22	83.83	63.26	66.93
RCK2, k-means	83.86	92.45	84.58	94.08	58.95	83.76	–	–
RCK2, hierarchical	86.11	95.50	89.06	95.29	62.08	85.64	61.28	71.14
RCK2, spectral	81.63	91.39	88.32	94.64	58.03	83.37	61.50	70.07
RCK3, k-means	83.66	92.59	84.13	92.96	58.60	83.28	–	–
RCK3, hierarchical	84.97	95.29	89.06	94.57	62.95	86.07	59.13	71.21
RCK3, spectral	81.16	91.56	88.32	94.73	55.83	83.20	59.26	71.00

Table: Proposed methods: Accuracy results using different kernels. The results are given in percentage. For each data set the best results are put in a framebox.

– best, – second best, – third best results

Conclusions

- Wikipedia kernel:
  - slightly outperforms the second baseline
  - reasons: some terms are not found in Wikipedia, unsuitable base feature selection (χ^2), distributional differences of terms in the corpora, etc.
- cluster kernels (HCK and RCK):
  - better performance compared to not data-dependent kernels
  - in many cases better results than that provided by other data-dependent kernels
  - max. improvement (compared to baseline results): using HCK/gHCK 13.5% accuracy improvement for COIL2/100
  - $\Rightarrow$ significantly outperform conventional kernels (linear, Gaussian)
  - work on a large variety of data sets
  - useful and efficient tools in many application domains where classification algorithms are needed

Publications

[Minier et al., 2006] Zsolt Minier, Zalán Bodó & Lehel Csató. Segmentation-based feature selection for text categorization. In Proceedings of the 2nd International Conference on Intelligent Computer Communication and Processing, pp. 53–59, 2006.

[Bodó et al., 2007] Zalán Bodó, Zsolt Minier & Lehel Csató. Text Categorization Experiments Using Wikipedia. In Proceedings of the conference Knowledge Engineering: Principles and Techniques, pp. 66–72, 2007.

[Minier et al., 2007] Zsolt Minier, Zalán Bodó & Lehel Csató. Wikipedia-based Kernels for Text Categorization. In Proceedings of the 9th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pp. 157–164, 2007.

[Csató & Bodó, 2008] Lehel Csató & Zalán Bodó. Neurális hálók és a gépi tanulás módszerei (Neural networks and machine learning methods). Presa Universitară Clujeană, Cluj-Napoca, 2008.

[Bodó, 2008] Zalán Bodó. Hierarchical Cluster Kernels For Supervised And Semi-Supervised Learning. In Proceedings of the 4nd International Conference on Intelligent Computer Communication and Processing, pp. 9–16, 2008.

[Bodó & Minier, 2008] Zalán Bodó & Zsolt Minier. On Supervised and Semi-Supervised K-Nearest Neighbor Algorithms. Presented at the 7th Joint Conference on Mathematics and Computer Science, Cluj-Napoca, Romania, 2008; appeared in STUDIA UNIV. BABEŞ-BOLYAI, INFORMATICA, Volume LIII, Number 2, Cluj-Napoca, 2008, pp. 79–92.

[Csató & Bodó, 2009] Lehel Csató & Zalán Bodó.
Decomposition Methods for Label Propagation, in Proceedings of the conference Knowledge Engineering: Principles and Techniques (KEPT 2009), Presa Universitară Clujeană, July 2–4, 2009. Special Issue of Studia Universitatis Babeș–Bolyai, Series Informatica. 2009, pp. 127–130.

[Bodó & Minier, 2009] Zalán Bodó & Zsolt Minier.
Semi-supervised Feature Selection with SVMS, in Proceedings of the conference Knowledge Engineering: Principles and Techniques (KEPT 2009), Presa Universitară Clujeană, July 2–4, 2009. Special Issue of Studia Universitatis Babeș–Bolyai, Series Informatica. 2009, pp. 159–162.

THANK YOU!