23rd International Conference on Computational Linguistics

Kernel Engineering for Fast and Easy Design of Natural Language Applications

Alessandro Moschitti
Department of Information Engineering and Computer Science
University of Trento
August 22, 2010
©2010, Alessandro Moschitti, all rights reserved

To order the CD of Coling 2010 and its Workshop Proceedings, please contact:

Chinese Information Processing Society of China
No.4, Southern Fourth Street
Haidian District, Beijing, 100190
China
Tel: +86-010-62562916
Fax: +86-010-62562916
cips@iscas.ac.cn
Tutorial Instructor

**Alessandro Moschitti:** DISI, University of Trento, Italy. E-mail: moschitti@disi.unitn.it

Alessandro is a professor at the Information Engineering and Computer Science Department of the University of Trento. In 2003, he obtained his PhD in Computer Science at the University of Rome “Tor Vergata” and between 2002 and 2004, he worked as an associate researcher in the University of Texas at Dallas for two years. His expertise concerns machine learning approaches to Natural Language Processing, Information Retrieval and Data Mining. In particular, he has recently devised innovative kernels within Support Vector and other kernel-based machines for advanced syntactic/semantic processing. He is author of more than 100 scientific articles published in the major conferences of different research communities, e.g. ACL, ICML, CIKM and ICDM. He has participated in several projects of the European Community (EC), e.g. LIVING-KNOWLEDGE 2008, PRESTOSPACE 2004, NAMIC 2000 and TREVI 1998 and in two US projects: MTBF 2008 (Con-Edison) and ARDA AQUAINT PROGRAM (IQAS 2002). He is currently project coordinator of the EC project, EternalS.
Outline

Previous work on the use of Machine Learning for Computational Linguistics has shown that most of the design effort is devoted to feature engineering. Indeed, the latter requires expertise, intuition and deep knowledge about the target problem to convert linguistic objects into attribute-value representations. Kernel Methods (KM) are powerful techniques, which can simplify data modeling by defining abstract representations and implicit feature spaces. More in particular, KM allow for: (a) directly using a similarity function between instances in learning algorithms, thus avoiding explicit feature design; and (b) implicitly defining huge feature spaces, e.g. structures can be represented in the substructure space.

In this tutorial, practical recipes to successfully use KM for target language applications will be presented: first, after an introduction to Support Vector Machines (explained from an application viewpoint), KM theory will be explained in a way that useful practical methods can be derived. Second, basic kernels, such as linear, polynomial, sequence and tree kernels will be presented, by focusing on the implementation, accuracy and efficiency perspectives. KM application to typical natural language tasks, e.g. text categorization, question and answer classification, semantic role labeling, textual entailment and so on, will be shown. The aim is to provide practical procedures for the selection and exploitation of the right kernel for the target task. Third, the SVM-Light-TK toolkit, which encodes several kernels in SVMs, will be illustrated along with the associated data structures and its practical use in NL tasks. Finally, the tutorial will illustrate how innovative and effective kernels can be engineered starting from basic kernels and using systematic data transformation. Such know-how allows for a very fast and accurate design of applications even if the underlying language phenomena and properties are still not very well understood, e.g. Arabic SRL or relation extraction between pairs of text fragments.
Kernel Engineering for Fast and Easy Design of Natural Language Applications

Alessandro Moschitti
Department of Information Engineering and Computer Science
University of Trento
Email: moschitti@disi.unitn.it

The 23rd International Conference on Computational Linguistics
August 22, 2010
Beijing, China

Schedule

- 14:00 - 15:30 First part
- 15:30 - 16:00 Coffee break
- 16:00 - 17:30 Second part
Outline (1)

- Motivation
- Kernel-Based Machines
  - Perceptron
  - Support Vector Machines
- Kernel Definition
  - Kernel Trick
  - Mercer’s conditions
  - Kernel operators
- Basic Kernels
  - Linear Kernel
  - Polynomial Kernel
  - Lexical Kernel

Outline (2)

- Structural Kernels
  - String and Word Sequence Kernels
  - Tree Kernels
    - Subtree, Syntactic, Partial Tree Kernels
- Applied Examples of Structural Kernels
  - Semantic Role Labeling (SRL)
  - Question Classification (QC)
  - SVM-Light-TK
  - Experiments in classroom with SRL and QC
  - Inspection of the input, output, and model files
Outline (3)

- Kernel Engineering
  - Structure Transformation
  - Syntactic Semantic Tree kernels
  - Kernel Combinations
  - Kernels on Object Pairs
  - Kernels for re-ranking
- Practical Question and Answer Classifier based on SVM-Light-TK
  - Combining Kernels
- Conclusion and Future Work

Motivation (1)

- Feature design most difficult aspect in designing a learning system
  - complex and difficult phase, e.g., structural feature representation:
  - deep knowledge and intuitions are required
  - design problems when the phenomenon is described by many features
Motivation (2)

- Kernel methods alleviate such problems
  - Structures represented in terms of substructures
  - High dimensional feature spaces
  - Implicit and abstract feature spaces
- Generate high number of features
  - Support Vector Machines “select” the relevant features
  - Automatic Feature engineering side-effect

Part I: Kernel Methods Theory
A simple classification problem: Text Categorization

Text Classification Problem

- Given:
  - a set of target categories: \( C = \{ C^1, \ldots, C^n \} \)
  - the set \( T \) of documents,
- define
  \[
  f : T \rightarrow 2^C
  \]
- VSM (Salton89')
  - Features are dimensions of a Vector Space.
  - Documents and Categories are vectors of feature weights.
  - \( d \) is assigned to \( C^i \) if \( \vec{d} \cdot \vec{C}^i > th \)
More in detail

- In Text Categorization documents are word vectors

\[
\Phi(d_x) = \tilde{x} = (0, ..., 1, ..., 0, ..., 1, ..., 0, ..., 1, ..., 0, ..., 1, ..., 0, ..., 1)
\]

- The dot product counts the number of features in common

\[
\tilde{x} \cdot \tilde{z} = (0, ..., 1, ..., 0, ..., 1, ..., 0, ..., 1, ..., 0, ..., 1, ..., 0, ..., 0)
\]

- This provides a sort of similarity

Linear Classifier

- The equation of a hyperplane is

\[
f(\tilde{x}) = \tilde{x} \cdot \tilde{w} + b = 0, \quad \tilde{x}, \tilde{w} \in \mathbb{R}^n, b \in \mathbb{R}
\]

- \(\tilde{x}\) is the vector representing the classifying example

- \(\tilde{w}\) is the gradient of the hyperplane

- The classification function is

\[
h(x) = \text{sign}(f(x))
\]
The main idea of Kernel Functions

- Mapping vectors in a space where they are linearly separable  $\vec{x} \rightarrow \phi(\vec{x})$

A mapping example

- Given two masses $m_1$ and $m_2$, one is constrained
- Apply a force $f_a$ to the mass $m_1$
- Experiments
  - Features $m_1$, $m_2$ and $f_a$
  - We want to learn a classifier that tells when a mass $m_1$ will get far away from $m_2$
  - If we consider the Gravitational Newton Law

$$f(m_1, m_2, r) = C \frac{m_1 m_2}{r^2}$$

- we need to find when $f(m_1, m_2, r) < f_a$
A mapping example (2)

\[ \bar{x} = (x_1, \ldots, x_n) \rightarrow \phi(\bar{x}) = (\phi_1(\bar{x}), \ldots, \phi_n(\bar{x})) \]

The gravitational law is not linear so we need to change space

\[ (f_a, m_1, m_2, r) \rightarrow (k, x, y, z) = (\ln f_a, \ln m_1, \ln m_2, \ln r) \]

As

\[ \ln f(m_1, m_2, r) = \ln C + \ln m_1 + \ln m_2 - 2 \ln r = c + x + y - 2z \]

We need the hyperplane

\[ \ln f_a - \ln m_1 - \ln m_2 + 2 \ln r - \ln C = 0 \]

\[ (\ln m_1, \ln m_2, -2 \ln r) \cdot (x, y, z) - \ln f_a + \ln C = 0, \]

we can decide without error if the mass will get far away or not

---

A kernel-based Machine Perceptron training

\[ \tilde{w}_0 \leftarrow \tilde{0}; b_0 \leftarrow 0; k \leftarrow 0; R \leftarrow \max_{i \in [\ell]} \| \bar{x}_i \| \]

do

for i = 1 to \ell

if \( y_i (\tilde{w}_k \cdot \bar{x}_j + b_k) \leq 0 \) then

\[ \tilde{w}_{k+1} = \tilde{w}_k + \eta y_i \bar{x}_j \]

\[ b_{k+1} = b_k + \eta y_i R^2 \]

k = k + 1

endif

endfor

while an error is found

return k,(\( \tilde{w}_k, b_k \))
Novikoff’s Theorem

Let \( S \) be a non-trivial training-set and let

\[
R = \max_{1 \leq i \leq l} \| x_i \|.
\]

Let us suppose there is a vector \( \mathbf{w}^* \), \( \| \mathbf{w}^* \| = 1 \) and

\[
y_i((\mathbf{w}^*, \mathbf{x}_i) + b^*) \approx \gamma, \quad i = 1, \ldots, l,
\]

with \( \gamma > 0 \). Then the maximum number of errors of the perceptron is:

\[
t^* = \left( \frac{2R}{\gamma} \right)^2.
\]
In each step of perceptron only training data is added with a certain weight

\[ \tilde{w} = \sum_{j=1}^{\ell} \alpha_j y_j \tilde{x}_j \]

So the classification function

\[ \text{sgn}(\tilde{w} \cdot \tilde{x} + b) = \text{sgn} \left( \sum_{j=1}^{\ell} \alpha_j y_j \tilde{x}_j \cdot \tilde{x} + b \right) \]

Note that data only appears in the scalar product

---

as well as the updating function

\[ \text{if } y_i \left( \sum_{j=1}^{\ell} \alpha_j y_j \tilde{x}_j \cdot \tilde{x}_i + b \right) \leq 0 \text{ then } \alpha_i = \alpha_i + \eta \]

The learning rate \( \eta \) only affects the re-scaling of the hyperplane, it does not affect the algorithm, so we can fix \( \eta = 1 \).
We can rewrite the classification function as

\[ h(x) = \text{sgn}(\mathbf{\tilde{w}}_\phi \cdot \phi(x) + b_\phi) = \text{sgn}(\sum_{j=1}^{\ell} \alpha_j y_j \phi(x_j) \cdot \phi(x) + b_\phi) = \text{sgn}(\sum_{j=1}^{\ell} \alpha_j y_j k(x_j, x) + b_\phi) \]

As well as the updating function

\[ \text{if } y \left( \sum_{j=1}^{\ell} \alpha_j y_j k(x_j, x) + b_\phi \right) \leq 0 \text{ allora } \alpha_i = \alpha_i + \eta \]

Support Vector Machines

- **Hard-margin SVMs**
- **Soft-margin SVMs**
Which hyperplane do we choose?

IDEA 1: Select the hyperplane with maximum margin
Support Vectors

Support Vector Machines

The margin is equal to \( \frac{2|k|}{\|w\|} \)
Support Vector Machines

The margin is equal to \( \frac{2|k|}{\|w\|} \).

We need to solve:

\[
\max \frac{2|k|}{\|w\|} \quad \text{subject to} \quad \begin{cases} 
\vec{w} \cdot \vec{x} + b \geq +k, & \text{if } \vec{x} \text{ is positive} \\
\vec{w} \cdot \vec{x} + b \leq -k, & \text{if } \vec{x} \text{ is negative}
\end{cases}
\]

There is a scale for which \( k = 1 \).

The problem transforms in:

\[
\max \frac{2}{\| \vec{w} \|} \quad \text{subject to} \quad \begin{cases} 
\vec{w} \cdot \vec{x} + b \geq +1, & \text{if } \vec{x} \text{ is positive} \\
\vec{w} \cdot \vec{x} + b \leq -1, & \text{if } \vec{x} \text{ is negative}
\end{cases}
\]
Final Formulation

\[
\max 2 \frac{\| \tilde{w} \|}{\| \tilde{w} \|} \Rightarrow \max \frac{2}{\| \tilde{w} \|} \Rightarrow \max \frac{2}{\| \tilde{w} \|} \Rightarrow \max \frac{2}{\| \tilde{w} \|} \Rightarrow
\]

\[
\tilde{w} \cdot \tilde{x}_i + b \geq +1, \ y_i = 1 \\
\tilde{w} \cdot \tilde{x}_i + b \leq -1, \ y_i = -1
\]

\[
\Rightarrow \min \frac{\| \tilde{w} \|}{2} \Rightarrow \min \frac{\| \tilde{w} \|}{2} \Rightarrow \min \frac{\| \tilde{w} \|}{2} \Rightarrow \min \frac{\| \tilde{w} \|}{2} \Rightarrow
\]

\[
y_i(\tilde{w} \cdot \tilde{x}_i + b) \geq 1 \\
y_i(\tilde{w} \cdot \tilde{x}_i + b) \geq 1
\]

Optimization Problem

- Optimal Hyperplane:
  - Minimize \( \tau(\tilde{w}) = \frac{1}{2} \| \tilde{w} \|^2 \)
  - Subject to \( y_i ((\tilde{w} \cdot \tilde{x}_i) + b) \geq 1, i = 1, \ldots, l \)

- The dual problem is simpler
Lagrangian Definition

Def. 2.24 Let $f(\bar{w})$, $h_i(\bar{w})$ and $g_i(\bar{w})$ be the objective function, the equality constraints and the inequality constraints (i.e. $\geq$) of an optimization problem, and let $L(\bar{w}, \bar{\alpha}, \bar{\beta})$ be its Lagrangian, defined as follows:

$$L(\bar{w}, \bar{\alpha}, \bar{\beta}) = f(\bar{w}) + \sum_{i=1}^{m} \alpha_i g_i(\bar{w}) + \sum_{i=1}^{l} \beta_i h_i(\bar{w})$$

Dual Optimization Problem

The Lagrangian dual problem of the above primal problem is

maximize $\theta(\bar{\alpha}, \bar{\beta})$

subject to $\bar{\alpha} \geq \bar{0}$

where $\theta(\bar{\alpha}, \bar{\beta}) = \inf_{\bar{w} \in W} L(\bar{w}, \bar{\alpha}, \bar{\beta})$
Dual Transformation

- Given the Lagrangian associated with our problem

\[ L(\overline{w}, b, \alpha) = \frac{1}{2} \overline{w} \cdot \overline{w} - \sum_{i=1}^{m} \alpha_i [y_i(\overline{w} \cdot \overline{x}_i + b) - 1] \]

- To solve the dual problem we need to evaluate:

\[ \theta(\overline{\alpha}, \overline{\beta}) = \inf_{\overline{w} \in W} \ L(\overline{w}, \overline{\alpha}, \overline{\beta}) \]

- Let us impose the derivatives to 0, with respect to \( \overline{w} \)

\[ \frac{\partial L(\overline{w}, b, \alpha)}{\partial \overline{w}} = \overline{w} - \sum_{i=1}^{m} y_i \alpha_i \overline{x}_i = \overline{0} \quad \Rightarrow \quad \overline{w} = \sum_{i=1}^{m} y_i \alpha_i \overline{x}_i \]

Dual Transformation (cont’d)

- and wrt \( b \)

\[ \frac{\partial L(\overline{w}, b, \alpha)}{\partial b} = \sum_{i=1}^{m} y_i \alpha_i = 0 \]

- Then we substituted them in the objective function

\[ L(\overline{w}, b, \alpha) = \frac{1}{2} \overline{w} \cdot \overline{w} - \sum_{i=1}^{m} \alpha_i [y_i(\overline{w} \cdot \overline{x}_i + b) - 1] = \]

\[ = \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \overline{x}_i \cdot \overline{x}_j - \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \overline{x}_i \cdot \overline{x}_j + \sum_{i=1}^{m} \alpha_i \]

\[ = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \overline{x}_i \cdot \overline{x}_j \]
The Final Dual Optimization Problem

$$\text{maximize} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \bar{x}_i \cdot \bar{x}_j$$

subject to \( \alpha_i \geq 0, \quad i = 1, \ldots, m \)
$$\sum_{i=1}^{m} y_i \alpha_i = 0$$

Khun-Tucker Theorem

- Necessary and sufficient conditions to optimality

$$\frac{\partial L(\bar{w}^*, \bar{\alpha}^*, \bar{\beta}^*)}{\partial \bar{w}} = \vec{0}$$
$$\frac{\partial L(\bar{w}^*, \bar{\alpha}^*, \bar{\beta}^*)}{\partial \bar{\alpha}} = \vec{0}$$
$$\frac{\partial b}{\partial \bar{\beta}} \left( \alpha_i^* g_i(\bar{w}^*) \right) = 0, \quad i = 1, \ldots, m$$
$$g_i(\bar{w}^*) \leq 0, \quad i = 1, \ldots, m$$
$$\alpha_i^* \geq 0, \quad i = 1, \ldots, m$$
**Properties coming from constraints**

- Lagrange constraints: \( \sum_{i=1}^{l} \alpha_i y_i = 0, \quad \hat{w} = \sum_{i=1}^{l} \alpha_i y_i \bar{x}_i \)
- Karush-Kuhn-Tucker constraints
  \[ \alpha_i \cdot [y_i (\bar{x}_i \cdot \hat{w} + b) - 1] = 0, \quad i = 1, ..., l \]
- Support Vectors have \( \alpha_i \), not null
- To evaluate \( b \), we can apply the following equation
  \[ b^* = -\frac{\hat{w}^+ \cdot \bar{x}^+ + \hat{w}^- \cdot \bar{x}^-}{2} \]

---

**Soft Margin SVMs**

\( \xi \), slack variables are added

Some errors are allowed but they should penalize the objective function
Soft Margin SVMs

The new constraints are

\[ y_i (\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i \]
\[ \forall \vec{x}_i \text{ where } \xi_i \geq 0 \]

The objective function penalizes the incorrect classified examples

\[ \min \frac{1}{2} \|\vec{w}\|^2 + C \sum_i \xi_i \]

\( C \) is the trade-off between margin and the error

---

Dual formulation

\[
\begin{align*}
\text{min} & \quad \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^{m} \xi_i^2 \\
y_i (\vec{w} \cdot \vec{x}_i + b) & \geq 1 - \xi_i, \quad \forall i = 1, \ldots, m \\
\xi_i & \geq 0, \quad i = 1, \ldots, m
\end{align*}
\]

\[ L(\vec{w}, b, \xi, \alpha) = \frac{1}{2} \|\vec{w}\|^2 + \frac{C}{2} \sum_{i=1}^{m} \xi_i^2 - \sum_{i=1}^{m} \alpha_i [y_i (\vec{w} \cdot \vec{x}_i + b) - 1 + \xi_i], \]

- By deriving wrt \( \vec{w}, \xi \) and \( b \)
Partial Derivatives

\[
\frac{\partial L(\vec{w}, b, \vec{\xi}, \vec{\alpha})}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^{m} y_i \alpha_i \vec{x}_i = \vec{0} \quad \Rightarrow \quad \vec{w} = \sum_{i=1}^{m} y_i \alpha_i \vec{x}_i
\]
\[
\frac{\partial L(\vec{w}, b, \vec{\xi}, \vec{\alpha})}{\partial \vec{\xi}} = C \vec{\xi} - \vec{\alpha} = \vec{0}
\]
\[
\frac{\partial L(\vec{w}, b, \vec{\xi}, \vec{\alpha})}{\partial b} = \sum_{i=1}^{m} y_i \alpha_i = 0
\]

Substitution in the objective function

\[
= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j + \frac{1}{2C} \vec{\alpha} \cdot \vec{\alpha} - \frac{1}{C} \vec{\alpha} \cdot \vec{\alpha} =
\]
\[
= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j - \frac{1}{2C} \vec{\alpha} \cdot \vec{\alpha} =
\]
\[
= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j (\vec{x}_i \cdot \vec{x}_j + \frac{1}{C} \delta_{ij}),
\]

\(\delta_{ij}\) of Kronecker
Final dual optimization problem

\[ \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j + \frac{1}{C} \delta_{ij}) \]

\[ \alpha_i \geq 0, \quad \forall i = 1, \ldots, m \]

\[ \sum_{i=1}^{m} y_i \alpha_i = 0 \]

Soft Margin Support Vector Machines

\[ \min \frac{1}{2} \| \tilde{w} \|^2 + C \sum_i \xi_i \quad y_i (\tilde{w} \cdot \tilde{x}_i + b) \geq 1 - \xi_i \quad \forall \tilde{x}_i \]

- \[ \xi_i \geq 0 \]

- The algorithm tries to keep \( \xi \) low and maximize the margin

- NB: The number of error is not directly minimized (NP-complete problem); the distances from the hyperplane are minimized

- If \( C \to \infty \), the solution tends to the one of the hard-margin algorithm

- Attention !!!: if \( C = 0 \) we get \( \| \tilde{w} \| = 0 \), since \( y_i b \geq 1 - \xi_i \), \( \forall \tilde{x}_i \)

- If \( C \) increases the number of error decreases. When \( C \) tends to infinite the number of errors must be 0, i.e. the hard-margin formulation
Robusteness of Soft vs. Hard Margin SVMs

In Soft Margin SVMs we maximize:

$$\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j + \frac{1}{C} \delta_{ij})$$

By using kernel functions we rewrite the problem as:

$$\begin{align*}
\text{maximize} & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j (k(\mathbf{u}_i, \mathbf{u}_j) + \frac{1}{C} \delta_{ij}) \\
\text{subject to} & \alpha_i \geq 0, \quad \forall i = 1, \ldots, m \\
& \sum_{i=1}^{m} y_i \alpha_i = 0
\end{align*}$$

Kernels in Support Vector Machines
Kernel Function Definition

Def. 2.26 A kernel is a function $k$, such that $\forall \vec{x}, \vec{z} \in X$

\[ k(\vec{x}, \vec{z}) = \phi(\vec{x}) \cdot \phi(\vec{z}) \]

where $\phi$ is a mapping from $X$ to an (inner product) feature space.

- Kernels are the product of mapping functions such as

\[ \vec{x} \in \mathbb{R}^n, \quad \phi(\vec{x}) = (\phi_1(\vec{x}), \phi_2(\vec{x}), \ldots, \phi_m(\vec{x})) \in \mathbb{R}^m \]

The Kernel Gram Matrix

- With KM-based learning, the sole information used from the training data set is the Kernel Gram Matrix

\[ K_{\text{training}} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \ldots & k(x_1, x_m) \\ k(x_2, x_1) & k(x_2, x_2) & \ldots & k(x_2, x_m) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_m, x_1) & k(x_m, x_2) & \ldots & k(x_m, x_m) \end{bmatrix} \]

- If the kernel is valid, $K$ is symmetric definite-positive.
Valid Kernels

Def. B.11 Eigen Values
Given a matrix \( A \in \mathbb{R}^m \times \mathbb{R}^n \), an eigenvalue \( \lambda \) and an eigenvector \( \bar{x} \in \mathbb{R}^n \) are such that

\[ A\bar{x} = \lambda \bar{x} \]

Def. B.12 Symmetric Matrix
A square matrix \( A \in \mathbb{R}^n \times \mathbb{R}^n \) is symmetric iff \( A_{ij} = A_{ji} \) for \( i \neq j \) and \( i = 1, \ldots, n \) and \( j = 1, \ldots, n \), i.e. iff \( A = A' \).

Def. B.13 Positive (Semi-) definite Matrix
A square matrix \( A \in \mathbb{R}^n \times \mathbb{R}^n \) is said to be positive (semi-) definite if its eigenvalues are all positive (non-negative).

Valid Kernels cont’d

Proposition 2.27 (Mercer’s conditions)
Let \( X \) be a finite input space with \( K(\bar{x}, \bar{z}) \) a symmetric function on \( X \). Then \( K(\bar{x}, \bar{z}) \) is a kernel function if and only if the matrix

\[ k(\bar{x}, \bar{z}) = \phi(\bar{x}) \cdot \phi(\bar{z}) \]

is positive semi-definite (has non-negative eigenvalues).

- If the matrix is positive semi-definite then we can find a mapping \( \phi \) implementing the kernel function
Mercer’s Theorem (finite space)

- Let us consider \( K = \left( K(\tilde{x}_i, \tilde{x}_j) \right)_{i,j=1}^n \)
- \( K \) symmetric \( \Rightarrow \exists V: K = V \Lambda V' \) for Takagi factorization of a complex-symmetric matrix, where:
  - \( \Lambda \) is the diagonal matrix of the eigenvalues \( \lambda_t \) of \( K \)
  - \( \tilde{v}_t = (v_{it})_{i=1}^n \) are the eigenvectors, i.e. the columns of \( V \)
- Let us assume lambda values non-negative
  \( \phi: \tilde{x}_i \rightarrow \left( \sqrt{\lambda_t} v_{it} \right)_{t=1}^n \in \mathbb{R}^n, i = 1, \ldots, n \)

Mercer’s Theorem (sufficient conditions)

- Therefore
  \[ \Phi(\tilde{x}_i) \cdot \Phi(\tilde{x}_j) = \sum_{t=1}^n \lambda_t v_{it} v_{jt} = (V \Lambda V')_{ij} = K_{ij} = K(\tilde{x}_i, \tilde{x}_j) \]
- which implies that \( K \) is a kernel function
Mercer’s Theorem (necessary conditions)

- Suppose we have negative eigenvalues $\lambda_s$ and eigenvectors $\mathbf{v}_s$ the following point

$$\mathbf{z} = \sum_{i=1}^{n} v_i \Phi(\mathbf{x}_i) = \sum_{i=1}^{n} v_i \left( \sqrt{\lambda_i} \mathbf{v}_i \right) = \mathbf{V} \Lambda' \mathbf{v}_s \hspace{1cm} \mathbf{x} = \mathbf{V} \Lambda' \mathbf{v}_s$$

- has the following norm:

$$\|\mathbf{z}\|^2 = \mathbf{z} \cdot \mathbf{z} = \mathbf{V} \Lambda' \mathbf{v}_s \cdot \mathbf{V} \Lambda' \mathbf{v}_s = \mathbf{v}_s' \mathbf{V} \Lambda' \mathbf{v}_s = \mathbf{v}_s' \mathbf{K} \mathbf{v}_s = \mathbf{v}_s' \mathbf{\lambda} \mathbf{v}_s = \lambda_s \|\mathbf{v}_s\|^2 < 0$$

this contradicts the geometry of the space.

Is it a valid kernel?

- It may not be a kernel so we can use $\mathbf{M}' \mathbf{M}$

**Proposition B.14** Let $\mathbf{A}$ be a symmetric matrix. Then $\mathbf{A}$ is positive (semi-) definite iff for any vector $\mathbf{x} \neq 0$

$$\mathbf{x}' \mathbf{A} \mathbf{x} > 0 \quad (\geq 0).$$

From the previous proposition it follows that: If we find a decomposition $\mathbf{A}$ in $\mathbf{M}' \mathbf{M}$, then $\mathbf{A}$ is semi-definite positive matrix as

$$\mathbf{x}' \mathbf{A} \mathbf{x} = \mathbf{x}' \mathbf{M}' \mathbf{M} \mathbf{x} = (\mathbf{M} \mathbf{x})'(\mathbf{M} \mathbf{x}) = \mathbf{M} \mathbf{x}' \mathbf{M} \mathbf{x} = ||\mathbf{M} \mathbf{x}||^2 \geq 0.$$
Valid Kernel operations

- $k(x,z) = k_1(x,z) + k_2(x,z)$
- $k(x,z) = k_1(x,z) \cdot k_2(x,z)$
- $k(x,z) = \alpha k_1(x,z)$
- $k(x,z) = f(x)f(z)$
- $k(x,z) = k_1(\phi(x), \phi(z))$
- $k(x,z) = x'Bz$

Basic Kernels for unstructured data

- Linear Kernel
- Polynomial Kernel
- Lexical kernel
- String Kernel
Linear Kernel

- In Text Categorization, documents are word vectors.
- The dot product \( \Phi(d_x) \cdot \Phi(d_z) \) counts the number of features in common.
- This provides a sort of similarity.

\[
\Phi(d_x) = \bar{x} = (0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1)
\]

- buy   acquisition     stocks          sell     market

\[
\Phi(d_z) = \bar{z} = (0, \ldots, 1, \ldots, 0, \ldots, 0, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 0, \ldots, 0, \ldots, 0)
\]

- buy   company            stocks          sell

Feature Conjunction (polynomial Kernel)

- The initial vectors are mapped in a higher space.
- More expressive, as \( (x_1 x_2) \) encodes 
  \( \text{Stock+Market} \ vs. \ \text{Downtown+Market} \) features.
- We can smartly compute the scalar product as

\[
\Phi(\bar{x}) \cdot \Phi(\bar{z}) =
\begin{align*}
  &= (x_1^2 + x_2^2 + \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1, 1, 1, 1, 1)
  \cdot (z_1^2 + z_2^2 + \sqrt{2}z_1z_2, \sqrt{2}z_1, \sqrt{2}z_2, 1, 1, 1, 1, 1) \\
  &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1x_2z_1z_2 + 2x_1z_1 + 2x_2z_2 + 1 \\
  &= (x_1z_1 + x_2z_2 + 1)^2 = (\bar{x} \cdot \bar{z} + 1)^2 = K_{\text{Poly}}(\bar{x}, \bar{z})
\end{align*}
\]
Document Similarity

\[ SK(d_1, d_2) = \sum_{w_1 \in d_1, w_2 \in d_2} s(w_1, w_2) \]

where \( s \) is any similarity function between words, e.g. WordNet [Basili et al., 2005] similarity or LSA [Cristianini et al., 2002].

Good results when training data is small.

Lexical Semantic Kernel [CoNLL 2005]

- The document similarity is the SK function:

- where \( s \) is any similarity function between words, e.g. WordNet [Basili et al., 2005] similarity or LSA [Cristianini et al., 2002]
- Good results when training data is small
Using character sequences

\[ \phi("bank") = \vec{x} = (0, ..., 1, ..., 0, ..., \text{..}, 1, 0, \text{..}, 1, 0, \text{..}, 0, \text{..}, 0, \text{..}, 1, 0, \text{..}, 0, \text{..}, 0) \]

| bank | ank | bnk | bk | b |

\[ \phi("rank") = \vec{z} = (1, ..., 0, ..., 1, ..., 0, ..., \text{..}, 0, 1, \text{..}, 0, \text{..}, 0, \text{..}, 0, \text{..}, 0, \text{..}, 0, \text{..}, 0) \]

| rank | ank | rnk | rk | r |

\[ \vec{x} \cdot \vec{z} \text{ counts the number of common substrings} \]

\[ \vec{x} \cdot \vec{z} = \phi("bank") \cdot \phi("rank") = k("bank", "rank") \]

String Kernel

- Given two strings, the number of matches between their substrings is evaluated
- E.g. Bank and Rank
  - B, a, n, k, Ba, Ban, Bank, Bk, an, ank, nk, ...
  - R, a, n, k, Ra, Ran, Rank, Rk, an, ank, nk, ...
- String kernel over sentences and texts
- Huge space but there are efficient algorithms
Formal Definition

\[ s = s_1, \ldots, s_{|s|} \]
\[ \vec{t} = (i_1, \ldots, i_{|u|}) \quad u = s[\vec{t}] \]
\[ \phi_u(s) = \sum_{\vec{t}: u = s[\vec{t}]} \lambda^{l(\vec{t})}, \text{ where } l(\vec{t}) = i_{|u|} - i_j + 1 \]
\[ K(s, t) = \sum_{u \in \Sigma^*} \phi_u(s) \cdot \phi_u(t) = \sum_{u \in \Sigma^*} \sum_{\vec{t}: u = s[\vec{t}]} \lambda^{l(\vec{t})} \sum_{\vec{t}: u = t[\vec{t}]} \lambda^{l(\vec{t})} = \]
\[ = \sum_{u \in \Sigma^*} \sum_{\vec{t}: u = s[\vec{t}]} \sum_{\vec{t}: u = t[\vec{t}]} \lambda^{l(\vec{t}) + l(\vec{t})}, \text{ where } \Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n \]

Kernel between Bank and Rank

B, a, n, k, Ba, Ban, Bank, an, ank, nk, Bn, Bnk, Bk and ak are the substrings of Bank.

R, a, n, k, Ra, Ran, Rank, an, ank, nk, Rn, Rnk, Rk and ak are the substrings of Rank.
An example of string kernel computation

- $\phi_a(\text{Bank}) = \phi_a(\text{Rank}) = \lambda^{(i_1-i_1+1)} = \lambda^{(2-2+1)} = \lambda,$
- $\phi_h(\text{Bank}) = \phi_h(\text{Rank}) = \lambda^{(i_1-i_1+1)} = \lambda^{(3-3+1)} = \lambda,$
- $\phi_k(\text{Bank}) = \phi_k(\text{Rank}) = \lambda^{(i_1-i_1+1)} = \lambda^{(4-4+1)} = \lambda,$
- $\phi_{an}(\text{Bank}) = \phi_{an}(\text{Rank}) = \lambda^{(i_2-i_1+1)} = \lambda^{(3-2+1)} = \lambda^3,$
- $\phi_{ank}(\text{Bank}) = \phi_{ank}(\text{Rank}) = \lambda^{(i_3-i_1+1)} = \lambda^{(4-2+1)} = \lambda^3,$
- $\phi_{nk}(\text{Bank}) = \phi_{nk}(\text{Rank}) = \lambda^{(i_2-i_1+1)} = \lambda^{(4-3+1)} = \lambda^2$
- $\phi_{ak}(\text{Bank}) = \phi_{ak}(\text{Rank}) = \lambda^{(i_2-i_1+1)} = \lambda^{(4-2+1)} = \lambda^3$

$K(\text{Bank}, \text{Rank}) = (\lambda, \lambda, \lambda, \lambda^2, \lambda^3, \lambda^2, \lambda^3)(\lambda, \lambda, \lambda, \lambda^2, \lambda^3, \lambda^2, \lambda^3)$

$= 3\lambda^2 + 2\lambda^4 + 2\lambda^6$

Efficient Evaluation

- **Dynamic Programming technique**
- **Evaluate the spectrum string kernels**
- **Substrings of size $p$**
- **Sum the contribution of the different spectra**
Efficient Evaluation

Given two sequences $s_1a$ and $s_2b$, we define:

$$D_p(|s_1|, |s_2|) = \sum_{i=1}^{s_1} \sum_{r=1}^{s_2} \lambda^{|s_1|-i+|s_2|-r} \times SK_{p-1}(s_1[1 : i], s_2[1 : r]),$$

$s_1[1 : i]$ and $s_2[1 : r]$ are their subsequences from 1 to $i$ and 1 to $r$.

$$SK_p(s_1a, s_2b) = \begin{cases} 
\lambda^2 \times D_p(|s_1|, |s_2|) & \text{if } a = b; \\
0 & \text{otherwise.}
\end{cases}$$

$D_p$ satisfies the recursive relation:

$$D_p(k, l) = SK_{p-1}(s_1[1 : k], s_2[1 : l]) + \lambda D_p(k, l - 1) + \\
+ \lambda D_p(k - 1, l) - \lambda^2 D_p(k - 1, l - 1)$$

---

An example: SK(“Gatta”, ”Cata”)

- First, evaluate the SK with size $p=1$, i.e. “a”, “a”, “t”, “t”, “a”, “a”
- Store this in the table

<table>
<thead>
<tr>
<th>$SK_{p=1}$</th>
<th>g</th>
<th>a</th>
<th>t</th>
<th>t</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
</tr>
<tr>
<td>t</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>$\lambda^2$</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
</tr>
</tbody>
</table>
Evaluating DP2

- Evaluate the weight of the string of size $p$ in case a character will be matched
- This is done by multiplying the double summation by the number of substrings of size $p-1$

$$D_p([s_1, s_2]) = \sum_{i=1}^{p-1} \sum_{r=1}^{p-1} \lambda^{s_1[i] + s_2[r]} \times SK_{p-1}(s_1[1 : i], s_2[1 : r])$$

Evaluating the Predictive DP on strings of size 2 (second row)

- Let’s consider substrings of size 2 and suppose that:
  - we have matched the first “a”
  - we will match the next character that we will add to the two strings
- We compute the weights of matches above at different string positions with some not-yet known character “?”
- If the match occurs immediately after “a” the weight will be $\lambda^{t+1} \times \lambda^{t+1} = \lambda^4$ and we store just $\lambda^2$ in the DP entry in [“a”, “a”]

<table>
<thead>
<tr>
<th>$DP_2$</th>
<th>g</th>
<th>a</th>
<th>t</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
<td>$\lambda^4$</td>
</tr>
<tr>
<td>t</td>
<td>0</td>
<td>$\lambda^4$</td>
<td>$\lambda^4 + \lambda^2$</td>
<td>$\lambda^6 + \lambda^3 + \lambda^2$</td>
</tr>
</tbody>
</table>
### Evaluating the DP wrt different positions (second row)

- If the match for “gatta” occurs after “t” the weight will be $\lambda^{t+2}$ ($\times \lambda^2$ = $\lambda^5$) since the substring for it will be with “a☐?" 
- We write such prediction in the entry ["a","t"] 
- Same rationale for a match after the second “t”: we have the substring “a☐☐?” (matching with “a?” from “catta”) for a weight of $\lambda^{3+1}$ ($\times \lambda^2$)

<table>
<thead>
<tr>
<th></th>
<th>g</th>
<th>a</th>
<th>t</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
<td>$\lambda^4$</td>
</tr>
<tr>
<td>t</td>
<td>0</td>
<td>$\lambda^5$</td>
<td>$\lambda^4 + \lambda^2$</td>
<td>$\lambda^5 + \lambda^3 + \lambda^2$</td>
</tr>
</tbody>
</table>

### Evaluating the DP wrt different positions (third row)

- If the match occurs after “t” of “cata”, the weight will be $\lambda^{2+t}$ ($\times \lambda^2$ = $\lambda^5$) since it will be with the string “a☐?”, with a weight of $\lambda^3$ 
- If the match occurs after “t” of both “gatta” and “cata”, there are two ways to compose substring of size two: “a☐?” with weight $\lambda^4$ or “t?” with weight $\lambda^2$ $\Rightarrow$ the total is $\lambda^2 + \lambda^4$

<table>
<thead>
<tr>
<th></th>
<th>g</th>
<th>a</th>
<th>t</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
<td>$\lambda^4$</td>
</tr>
<tr>
<td>t</td>
<td>0</td>
<td>$\lambda^4$</td>
<td>$\lambda^4 + \lambda^2$</td>
<td>$\lambda^5 + \lambda^3 + \lambda^2$</td>
</tr>
</tbody>
</table>
Evaluating the DP wrt different positions (third row)

- The final case is a match after the last “t” of both “cat” and “gatta”
- There are three possible substrings of “gatta”:
  - “☐☐☐”, “☐☐?”,”☐?” for “gatta” with weight $\lambda^3$, $\lambda^2$ or $\lambda$, respectively.
- There are two possible substrings of “cata”
  - “☐☐?”,”☐?” with weight $\lambda^2$ and $\lambda$
  - Their match gives weights: $\lambda^5$, $\lambda^3$, $\lambda^2$ ⇒ by summing: $\lambda^5 + \lambda^3 + \lambda^2$

|      | DP
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>a</td>
<td>0 $\lambda^2$ $\lambda^3$ $\lambda^4$</td>
</tr>
<tr>
<td>t</td>
<td>0 $\lambda^3$ $\lambda^4 + \lambda^2$ $\lambda^5 + \lambda^3 + \lambda^2$</td>
</tr>
</tbody>
</table>

Evaluating SK of size 2 using DP2

$SK_p(s_1a,s_2b) = \begin{cases} 
\lambda^2 \times D_p(|s_1|,|s_2|) & \text{if } a = b, \\
0 & \text{otherwise.} 
\end{cases}$

|      | DP
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>a</td>
<td>0 $\lambda^2$ $\lambda^3$ $\lambda^4$</td>
</tr>
<tr>
<td>t</td>
<td>0 $\lambda^3$ $\lambda^4 + \lambda^2$ $\lambda^5 + \lambda^3 + \lambda^2$</td>
</tr>
<tr>
<td>a</td>
<td>0 0 0 0 $\lambda^4$</td>
</tr>
</tbody>
</table>

- The number (weight) of substrings of size 2 between “gat” and “cat” is $\lambda^4 = \lambda^2$ ($["a","a"]$ entry of DP) x $\lambda^2$(cost of one character), where $a$ = “t” and $b$ = “t”.
- Between “gatta” and “cata” is $\lambda^7 + \lambda^5 + \lambda^4$, i.e the matches of “☐☐☐a”, “☐☐ta”, “ta” with “☐☐a” and “ta”.

38
Tree kernels

- Subtree, Subset Tree, Partial Tree kernels
- Efficient computation

Example of a parse tree

“John delivers a talk in Rome”
The Syntactic Tree Kernel (STK)  
[Collins and Duffy, 2002]  

The overall fragment set
The overall fragment set

Explicit kernel space

\[ \phi(T_x) = \bar{x} = (0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0) \]

\[ \phi(T_z) = \bar{z} = (1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 0) \]

\[ \bar{x} \cdot \bar{z} \] counts the number of common substructures
Efficient evaluation of the scalar product

\[ \tilde{x} \cdot \tilde{z} = \phi(T_x) \cdot \phi(T_z) = K(T_x, T_z) = \]
\[ = \sum_{n_x \in T_x} \sum_{n_z \in T_z} \Delta(n_x, n_z) \]

- [Collins and Duffy, ACL 2002] evaluate \( \Delta \) in \( O(n^2) \):

\[ \Delta(n_x, n_z) = 0, \text{ if the productions are different} \]
\[ \Delta(n_x, n_z) = 1, \text{ if pre-terminals else} \]
\[ \Delta(n_x, n_z) = \prod_{j=1}^{nc(n_x)} (1 + \Delta(ch(n_x, j), ch(n_z, j))) \]
**Other Adjustments**

- **Decay factor**
  \[
  \Delta(n_x, n_z) = \lambda, \quad \text{if pre-terminals else}
  \]
  \[
  \Delta(n_x, n_z) = \lambda \prod_{j=1}^{nc(n_x)} (1 + \Delta(ch(n_x, j), ch(n_z, j)))
  \]

- **Normalization**
  \[
  K'(T_x, T_z) = \frac{K(T_x, T_z)}{\sqrt{K(T_x, T_x) \times K(T_z, T_z)}}
  \]

**SubTree (ST) Kernel** [Vishwanathan and Smola, 2002]

![SubTree Diagram]

- VP
- NP
- NP
- V
- D
- N
- a
- talk
- delivers
- a
- talk
- a
- talk
- delivers
- a
- talk
Given the equation for the SST kernel

\[ \Delta(n_x, n_z) = 0, \text{ if the productions are different else} \]
\[ \Delta(n_x, n_z) = 1, \text{ if pre-terminals else} \]
\[ \Delta(n_x, n_z) = \prod_{j=1}^{nc(n_x)} (1 + \Delta(ch(n_x, j), ch(n_z, j))) \]
Fast Evaluation of STK [Moschitti, EACL 2006]

\[ K(T_x, T_z) = \sum_{(n_x, n_z) \in NP} \Delta(n_x, n_z) \]

\[ NP = \left\{ (n_x, n_z) \in T_x \times T_z : \Delta(n_x, n_z) \neq 0 \right\} = \]
\[ = \left\{ (n_x, n_z) \in T_x \times T_z : P(n_x) = P(n_z) \right\}, \]

where \( P(n_x) \) and \( P(n_z) \) are the production rules used at nodes \( n_x \) and \( n_z \).

---

```plaintext
function Evaluate_Pair_Set(Tree T1, T2) returns NODE_PAIR_SET:
    LIST L1, L2:
    NODE_PAIR_SET NP:
    begin
        L1 = T1.ordered_list;
        L2 = T2.ordered_list; /*the lists were sorted at loading time*/
        n1 = extract(L1); /*get the head element and*/
        n2 = extract(L2); /*remove it from the list*/
        while (n1 and n2 are not NULL)
            if (production_of(n1) > production_of(n2))
                then n2 = extract(L2);
            else if (production_of(n1) < production_of(n2))
                then n1 = extract(L1);
            else
                while (production_of(n1) == production_of(n2))
                    while (production_of(n1) == production_of(n2))
                        add((n1, n2), NP);
                        n2 = get_next_elem(L2); /*get the head element and move the pointer to the next element*/
                    end
                    n1 = extract(L1);
                    reset(L2); /*set the pointer at the first element*/
                end
    end
return NP;
end
```
Observations

- We order the production rules used in $T_x$ and $T_z$, at loading time.
- At learning time we may evaluate NP in $|T_x| + |T_z|$ running time.
- If $T_x$ and $T_z$ are generated by only one production rule $\Rightarrow O(|T_x| \times |T_z|)$…Very Unlikely!!!!
Labeled Ordered Tree Kernel

- SST satisfies the constraint “remove 0 or all children at a time”.
- If we relax such constraint we get more general substructures [Kashima and Koyanagi, 2002]

Weighting Problems

- Both matched pairs give the same contribution.
- Gap based weighting is needed.
- A novel efficient evaluation has to be defined
Partial Trees, [Moschitti, ECML 2006]

- SST + String Kernel with weighted gaps on Nodes’ children

Partial Tree Kernel

- if the node labels of \( n_1 \) and \( n_2 \) are different then 
  \( \Delta(n_1, n_2) = 0 \);
- else
  \[ 
  \Delta(n_1, n_2) = 1 + \sum_{J_1, J_2, t(J_1) = t(J_2)} \prod_{i=1}^{t(J_1)} \Delta(c_{n_1}[J_{1i}], c_{n_2}[J_{2i}]) 
  \]

- By adding two decay factors we obtain:

\[
\mu \left( \lambda^2 + \sum_{J_1, J_2, t(J_1) = t(J_2)} \lambda^{d(J_1) + d(J_2)} \prod_{i=1}^{t(J_1)} \Delta(c_{n_1}[J_{1i}], c_{n_2}[J_{2i}]) \right)
\]
In [Taylor and Cristianini, 2004 book], sequence kernels with weighted gaps are factorized with respect to different subsequence sizes.

We treat children as sequences and apply the same theory

\[ \Delta(n_1, n_2) = \mu (\lambda^2 + \sum_{p=1}^{lm} \Delta_p(c_{n_1}, c_{n_2})) , \]

Given the two child sequences \( s_1a = c_{n_1} \) and \( s_2b = c_{n_2} \) (\( a \) and \( b \) are the last children), \( \Delta_p(s_1a, s_2b) = D_p \)

\[ \Delta(a, b) \times \sum_{i=1}^{s_1} \sum_{r=1}^{s_2} \lambda^{s_1-i-s_2-r} \times \Delta_{p-1}(s_1[1 : i], s_2[1 : r]) \]

---

**Efficient Evaluation (2)**

\[ \Delta_p(s_1a, s_2b) = \begin{cases} \Delta(a, b)D_p(|s_1|, |s_2|) & \text{if } a = b; \\ 0 & \text{otherwise.} \end{cases} \]

Note that \( D_p \) satisfies the recursive relation:

\[ D_p(k, l) = \Delta_{p-1}(s_1[1 : k], s_2[1 : l]) + \lambda D_p(k, l - 1) + \lambda D_p(k - 1, l) + \lambda^2 D_p(k - 1, l - 1). \]

The complexity of finding the subsequences is \( O(p|s_1||s_2|) \)

Therefore the overall complexity is \( O(p^2\rho|N_T_1||N_T_2|) \)

where \( \rho \) is the maximum branching factor (\( p = \rho \))
Running Time of Tree Kernel Functions

SVM-light-TK Software

- Encodes ST, SST and combination kernels in SVM-light [Joachims, 1999]
- Available at http://dit.unitn.it/~moschitt/
- Tree forests, vector sets
- The new SVM-Light-TK toolkit will be released asap
Data Format

- "What does Html stand for?"
- 1 | BT | (SBARQ (WHNP (WP What))(SQ (AUX does)(NP (NNP S.O.S.))(VP (VB stand)(PP (IN for))))( . ?))
- | BT | (BOW (What *) (does *)(S.O.S. *)(stand *)(for *)(? *))
- | BT | (BOP (WP *)(AUX *)(NNP *)(VB *)(IN *)(. *))
- | BT | (PAS (ARG0 (R-A1 (What *)))(ARG1 (A1 (S.O.S. NNP)))(ARG2 (rel stand)))

Basic Commands

- Training and classification
  - ./svm_learn -t 5 -C T train.dat model
  - ./svm_classify test.dat model
- Learning with a vector sequence
  - ./svm_learn -t 5 -C V train.dat model
- Learning with the sum of vector and kernel sequences
  - ./svm_learn -t 5 -C + train.dat model
Part II: Kernel Methods for Practical Applications

Kernel Engineering approaches

- Basic Combinations
- *Canonical Mappings*, e.g. object transformations
- *Merging of Kernels*
Kernel Combinations an example

\[ K_p^3 \] polynomial kernel of flat features

\( K_{\text{Tree}} \) Tree kernel

- Kernel Combinations:

\[
K_{\text{Tree}^+P} = \gamma \times K_{\text{Tree}} + K_p^3, \\
K_{\text{Tree}^xP} = K_{\text{Tree}} \times K_p^3
\]

Object Transformation [Moschitti et al, CLJ 2008]

- \( K(O_1, O_2) = \phi(O_1) \cdot \phi(O_2) = \phi_E(\phi_M(O_1)) \cdot \phi_E(\phi_M(O_2)) \)
  
  \[ = \phi_E(S_1) \cdot \phi_E(S_2) = K_E(S_1, S_2) \]

- **Canonical Mapping**, \( \phi_M() \)
  - object transformation,
  - e.g. a syntactic parse tree into a verb subcategorization frame tree.

- **Feature Extraction**, \( \phi_E() \)
  - maps the canonical structure in all its fragments
  - different fragment spaces, e.g. ST, SST and PT.
Predicate Argument Classification

- In an event:
  - target words describe relation among different entities
  - the participants are often seen as predicate's arguments.

- Example:
  Paul gives a talk in Rome

.Predicate Argument Classification

- In an event:
  - target words describe relation among different entities
  - the participants are often seen as predicate's arguments.

- Example:
  [Arg0 Paul] [predicate gives] [Arg1 a talk] [ArgM in Rome]
Predicate-Argument Feature Representation

Given a sentence, a predicate $p$:
1. Derive the sentence parse tree
2. For each node pair $<N_p, N_x>$
   a. Extract a feature representation set $F$
   b. If $N_x$ exactly covers the Arg-$i$, $F$ is one of its positive examples
   c. $F$ is a negative example otherwise

Vector Representation for the linear kernel
Kernel Engineering: Tree Tailoring

PAT Kernel [Moschitti, ACL 2004]

- Given the sentence:
  \[
  \text{[Arg0 Paul] [\text{predicate} delivers] [Arg1 a talk] [ArgM in formal Style]}
  \]

- These are Semantic Structures
In other words we consider…

Sub-Categorization Kernel (SCF)
[Moschitti, ACL 2004]
Experiments on Gold Standard Trees

- PropBank and PennTree bank
  - about 53,700 sentences
  - Sections from 2 to 21 train., 23 test., 1 and 22 dev.
  - Arguments from Arg0 to Arg5, ArgA and ArgM for a total of 122,774 and 7,359

- FrameNet and Collins' automatic trees
  - 24,558 sentences from the 40 frames of Senseval 3
  - 18 roles (same names are mapped together)
  - Only verbs
  - 70% for training and 30% for testing

Argument Classification with Poly Kernel

![Graph showing accuracy comparison between FrameNet and PropBank](image)
### PropBank Results

#### Argument Classification on PAT using different Tree Fragment Extractor

<table>
<thead>
<tr>
<th>Args</th>
<th>P3</th>
<th>PAT</th>
<th>PAT+P</th>
<th>PAT×P</th>
<th>SCF+P</th>
<th>SCF×P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arg0</td>
<td>90.8</td>
<td>88.3</td>
<td>92.6</td>
<td>90.5</td>
<td>94.6</td>
<td>94.7</td>
</tr>
<tr>
<td>Arg1</td>
<td>91.1</td>
<td>87.4</td>
<td>91.9</td>
<td>91.2</td>
<td>92.9</td>
<td>94.1</td>
</tr>
<tr>
<td>Arg2</td>
<td>80.0</td>
<td>68.5</td>
<td>77.5</td>
<td>74.7</td>
<td>77.4</td>
<td>82.0</td>
</tr>
<tr>
<td>Arg3</td>
<td>57.9</td>
<td>56.5</td>
<td>55.6</td>
<td>49.7</td>
<td>56.2</td>
<td>56.4</td>
</tr>
<tr>
<td>Arg4</td>
<td>70.5</td>
<td>68.7</td>
<td>71.2</td>
<td>62.7</td>
<td>69.6</td>
<td>71.1</td>
</tr>
<tr>
<td>ArgM</td>
<td>95.4</td>
<td>94.1</td>
<td>96.2</td>
<td>96.2</td>
<td>96.1</td>
<td>96.3</td>
</tr>
</tbody>
</table>

| Global Accuracy | 90.5 | 88.7 | 91.3 | 90.4 | 92.4 | 93.2 |

---

### Argument Classification on PAT using different Tree Fragment Extractor

![Graph showing accuracy over training data]
FrameNet Results

<table>
<thead>
<tr>
<th>Roles</th>
<th>P3</th>
<th>PAF</th>
<th>PAF+P</th>
<th>PAF×P</th>
<th>SCF+P</th>
<th>SCF×P</th>
</tr>
</thead>
<tbody>
<tr>
<td>agent</td>
<td>92.0</td>
<td>88.5</td>
<td>91.7</td>
<td>91.3</td>
<td>93.1</td>
<td>93.9</td>
</tr>
<tr>
<td>cause</td>
<td>59.7</td>
<td>16.1</td>
<td>41.6</td>
<td>27.7</td>
<td>42.6</td>
<td>57.3</td>
</tr>
<tr>
<td>degree</td>
<td>74.9</td>
<td>68.6</td>
<td>71.4</td>
<td>57.8</td>
<td>68.5</td>
<td>60.9</td>
</tr>
<tr>
<td>depictive</td>
<td>52.6</td>
<td>29.7</td>
<td>51.0</td>
<td>28.6</td>
<td>46.8</td>
<td>37.6</td>
</tr>
<tr>
<td>duration</td>
<td>45.8</td>
<td>52.1</td>
<td>40.9</td>
<td>29.0</td>
<td>31.8</td>
<td>41.8</td>
</tr>
<tr>
<td>goal</td>
<td>85.9</td>
<td>78.6</td>
<td>85.3</td>
<td>82.8</td>
<td>84.0</td>
<td>85.3</td>
</tr>
<tr>
<td>instrument</td>
<td>67.9</td>
<td>46.8</td>
<td>62.8</td>
<td>55.8</td>
<td>59.6</td>
<td>64.1</td>
</tr>
<tr>
<td>manner</td>
<td>81.0</td>
<td>81.9</td>
<td>81.2</td>
<td>78.6</td>
<td>77.8</td>
<td>77.8</td>
</tr>
<tr>
<td>Global Acc. (18 roles)</td>
<td><strong>85.2</strong></td>
<td><strong>79.5</strong></td>
<td><strong>84.6</strong></td>
<td><strong>81.6</strong></td>
<td><strong>83.8</strong></td>
<td><strong>84.2</strong></td>
</tr>
</tbody>
</table>

- ProbBank arguments vs. Semantic Roles

Kernel Engineering: Node marking
Marking Boundary nodes

A) PAF+
   VP
   V NP
   D N
   delivers a talk

B) PAF-
   VP
   V NP
   D N
   delivers talk

MPAF+
   VP
   V NP
   N-B
   delivers a talk

MPAF-
   VP
   V NP
   N-B
   delivers talk

Node Marking Effect

C) common PAF features
   VP
   V NP VP V N
   delivers V NP delivers talk

D) common MPAF features
   V
   delivers
Different tailoring and marking

Experiments

- PropBank and PennTree bank
  - about 53,700 sentences
  - Charniak trees from CoNLL 2005
- Boundary detection:
  - Section 2 training
  - Section 24 testing
  - PAF and MPAF
Number of examples/nodes of Section 2

<table>
<thead>
<tr>
<th></th>
<th>Section 2</th>
<th></th>
<th></th>
<th>Section 24</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pos</td>
<td>neg</td>
<td>tot</td>
<td>pos</td>
<td>neg</td>
<td>tot</td>
</tr>
<tr>
<td>Internal</td>
<td>11,847</td>
<td>71,126</td>
<td>82,973</td>
<td>7,525</td>
<td>50,123</td>
<td>57,648</td>
</tr>
<tr>
<td>Pre-terminal</td>
<td>894</td>
<td>114,052</td>
<td>114,946</td>
<td>709</td>
<td>80,366</td>
<td>81,075</td>
</tr>
<tr>
<td>Both</td>
<td>12,741</td>
<td>185,178</td>
<td>197,919</td>
<td>8,234</td>
<td>130,489</td>
<td>138,723</td>
</tr>
</tbody>
</table>

Predicate Argument Feature (PAF) vs. Marked PAF (MPAF) [Moschitti et al, ACL-ws-2005]

<table>
<thead>
<tr>
<th>Tagging strategy</th>
<th>CPU$_{time}$</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAF</td>
<td>5,179.18</td>
<td>75.24</td>
</tr>
<tr>
<td>MPAF</td>
<td>3,131.56</td>
<td>82.07</td>
</tr>
</tbody>
</table>
Merging of Kernels [ECIR 2007]:
Question/Answer Classification

- Syntactic/Semantic Tree Kernel
- Kernel Combinations
- Experiments

Merging of Kernels [Bloehdorn & Moschitti, ECIR 2007 & CIKM 2007]

Definition 4 (Tree Fragment Similarity Kernel). For two tree fragments $f_1, f_2 \in \mathcal{F}$, we define the Tree Fragment Similarity Kernel as:

$$\kappa_{\mathcal{F}}(f_1, f_2) = \text{comp}(f_1, f_2) \prod_{t=1}^{\text{nt}(f_1)} \kappa_{S}(f_1(t), f_2(t))$$

$$\kappa_T(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$$

where $\Delta(n_1, n_2) = \sum_{i=1}^{\vert \mathcal{F} \vert} \sum_{j=1}^{\vert \mathcal{F} \vert} I_i(n_1)I_j(n_2)\kappa_{\mathcal{F}}(f_i, f_j)$. 
Merging of Kernels

\[ \kappa_T(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2) \]

where \( \Delta(n_1, n_2) = \sum_{i=1}^{|F|} \sum_{j=1}^{|F|} I_i(n_1)I_j(n_2)\kappa_F(f_i, f_j). \)

Delta Evaluation is very simple

0. if \( n_1 \) and \( n_2 \) are pre-terminals and \( \text{label}(n_1) = \text{label}(n_2) \) then \( \Delta(n_1, n_2) = \lambda \kappa_S(ch_{n_1}^l, ch_{n_2}^l) \),
1. if the productions at \( n_1 \) and \( n_2 \) are different then \( \Delta(n_1, n_2) = 0; \)
2. \( \Delta(n_1, n_2) = \lambda \),
3. \( \Delta(n_1, n_2) = \lambda \prod_{j=1}^{n_c(n_1)} (1 + \Delta(ch_{n_1}^j, ch_{n_2}^j)) \).
Question Classification

- **Definition**: What does HTML stand for?
- **Description**: What's the final line in the Edgar Allan Poe poem "The Raven"?
- **Entity**: What foods can cause allergic reaction in people?
- **Human**: Who won the Nobel Peace Prize in 1992?
- **Location**: Where is the Statue of Liberty?
- **Manner**: How did Bob Marley die?
- **Numeric**: When was Martin Luther King Jr. born?
- **Organization**: What company makes Bentley cars?

Question Classifier based on Tree Kernels

- Question dataset (http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/) [Lin and Roth, 2005])
  - Distributed on 6 categories: Abbreviations, Descriptions, Entity, Human, Location, and Numeric.
- Fixed split 5500 training and 500 test questions
- Cross-validation (10-folds)
- Using the whole question parse trees
  - Constituent parsing
  - Example
    
    “What is an offer of direct stock purchase plan?”
Kernels

- BOW, POS are obtained with a simple tree, e.g.

- PT (parse tree)
- PAS (predicate argument structure)
## Question classification

<table>
<thead>
<tr>
<th>Features</th>
<th>Accuracy (UIUC)</th>
<th>Accuracy (c.v.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PT</td>
<td>90.4</td>
<td>84.8±1.4</td>
</tr>
<tr>
<td>BOW</td>
<td>90.6</td>
<td>84.7±1.4</td>
</tr>
<tr>
<td>PAS</td>
<td>34.2</td>
<td>43.0±2.2</td>
</tr>
<tr>
<td>POS</td>
<td>26.4</td>
<td>32.4±2.5</td>
</tr>
<tr>
<td><strong>PT+BOW</strong></td>
<td><strong>91.8</strong></td>
<td><strong>86.1±1.3</strong></td>
</tr>
<tr>
<td>PT+BOW+POS</td>
<td>91.8</td>
<td>84.7±1.7</td>
</tr>
<tr>
<td>PAS+BOW</td>
<td>90.0</td>
<td>82.1±1.5</td>
</tr>
<tr>
<td>PAS+BOW+POS</td>
<td>88.8</td>
<td>81.0±1.7</td>
</tr>
</tbody>
</table>

## Similarity based on WordNet

Inverted Path Length:

\[
s_{ipl}(c_1, c_2) = \frac{1}{(1 + d(c_1, c_2))^\alpha}
\]

Wu & Palmer:

\[
s_{wup}(c_1, c_2) = \frac{2 \text{dep}(\text{iso}(c_1, c_2))}{d(c_1, \text{iso}(c_1, c_2)) + d(c_2, \text{iso}(c_1, c_2)) + 2 \text{dep}(\text{iso}(c_1, c_2))}
\]

Resnik:

\[
s_{res}(c_1, c_2) = -\log P(\text{iso}(c_1, c_2))
\]

Lin:

\[
s_{lin}(c_1, c_2) = \frac{2 \log P(\text{iso}(c_1, c_2))}{\log P(c_1) + \log P(c_2)}
\]
Question Classification with S/STK

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$ parameter</td>
<td>0.4</td>
</tr>
<tr>
<td>linear (bow)</td>
<td>0.905</td>
</tr>
<tr>
<td>string matching</td>
<td>0.890</td>
</tr>
<tr>
<td>full</td>
<td>0.904</td>
</tr>
<tr>
<td>full-ic</td>
<td>0.908</td>
</tr>
<tr>
<td>path-1</td>
<td>0.906</td>
</tr>
<tr>
<td>path-2</td>
<td>0.896</td>
</tr>
<tr>
<td>lin</td>
<td>0.908</td>
</tr>
<tr>
<td>wup</td>
<td>0.908</td>
</tr>
</tbody>
</table>

Multiple Kernel Combinations
**TASK: Question/Answer Classification [Moschitti, CIKM 2008]**

- The classifier detects if a pair (question and answer) is correct or not
- A representation for the pair is needed
- The classifier can be used to re-rank the output of a basic QA system

**Dataset 2: TREC data**

- 138 TREC 2001 test questions labeled as “description”
- 2,256 sentences, extracted from the best ranked paragraphs (using a basic QA system based on Lucene search engine on TREC dataset)
- 216 of which labeled as correct by one annotator
Dataset 2: TREC data

- 138 TREC 2001 test questions labeled as “description”

  A question is linked to many answers: all its derived pairs cannot be shared by training and test sets

- 216 of which labeled as correct by one annotator

Bags of words (BOW) and POS-tags (POS)

- To save time, apply STK to these trees:

  ![Diagram of trees with POS tags]

  BOX
  
  What is an offer of ...
  
  WHNP VBZ DT NN IN ...
  
  * * * * *
Word and POS Sequences

- What is an offer of…? (word sequence, WSK)
  ➔ What_is_offer
  ➔ What_is

- WHNP VBZ DT NN IN…(POS sequence, POSSK)
  ➔ WHNP_VBZ_NN
  ➔ WHNP_NN_IN

Syntactic Parse Trees (PT)
Predicate Argument Structure for Partial Tree Kernel (PAS\textsubscript{PTK})

- [ARG1 Antigens] were [AM–TMP originally] [rel defined] [ARG2 as non-self molecules].
- [ARG0 Researchers] [rel describe] [ARG1 antigens][ARG2 as foreign molecules] [ARGM–LOC in the body]

Kernels and Combinations

- Exploiting the property: $k(x,z) = k_1(x,z) + k_2(x,z)$
- BOW, POS, WSK, POSSK, PT, PAS\textsubscript{PTK}
  $\Rightarrow$ BOW+POS, BOW+PT, PT+POS, ...
Results on TREC Data
(5 folds cross validation)

![Chart showing F1-measure for different kernel types]

Results on TREC Data
(5 folds cross validation)

![Chart showing F1-measure for different kernel types]
Results on TREC Data
(5 folds cross validation)

<table>
<thead>
<tr>
<th>Kernel Type</th>
<th>F1-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOW</td>
<td>20</td>
</tr>
<tr>
<td>BOW+POS</td>
<td>22</td>
</tr>
<tr>
<td>BOW+POS+PT</td>
<td>24</td>
</tr>
<tr>
<td>BOW+POS+PT+PAS_SSTK</td>
<td>26</td>
</tr>
<tr>
<td>BOW+POS+PT+PAS_PTK</td>
<td>28</td>
</tr>
</tbody>
</table>

Results on TREC Data
(5 folds cross validation)

<table>
<thead>
<tr>
<th>Kernel Type</th>
<th>F1-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOW</td>
<td>20</td>
</tr>
<tr>
<td>BOW+POS</td>
<td>22</td>
</tr>
<tr>
<td>BOW+POS+PT</td>
<td>24</td>
</tr>
<tr>
<td>BOW+POS+PT+PAS_SSTK</td>
<td>26</td>
</tr>
<tr>
<td>BOW+POS+PT+PAS_PTK</td>
<td>28</td>
</tr>
</tbody>
</table>

75
Results on TREC Data
(5 folds cross validation)
Results on TREC Data
(5 folds cross validation)

F1-measure

Kernel Type

BOW ≈ 24
POSSK+STK+PAS_PTK ≈ 39
⇒ 62% of improvement

Kernels for Re-ranking
Re-ranking Framework

- Local classifier generates the most likely set of hypotheses.
- These are used to build annotation pairs, \( \langle h^i, h^j \rangle \).
  - positive instances if \( h^i \) more correct than \( h^j \).
- A binary classifier decides if \( h^i \) is more accurate than \( h^j \).
- Each candidate annotation \( h^i \) is described by a structural representation
Syntactic Parsing Re-ranking

- Pairs of parse trees (Collins and Duffy, 2002)

Re-ranking concept labeling
[Dinarelli et al, 2009]

- I have a problem with my monitor
  
  $h^1$: I NULL have NULL a NULL problem PROBLEM-B with NULL my NULL monitor HW-B

  $h^2$: I NULL have NULL a NULL problem HW-B with NULL my NULL monitor
Flat tree representation (cross-language structure)

Multilevel Tree
Enriched Multilevel Tree

- FST CER from 23.2 to 16.01

Re-ranking for Named-Entity Recognition [Vien et al, 2010]

- CRF F1 from 84.86 to 88.16
Re-ranking Predicate Argument Structures
[Moschitti et al, CoNLL 2006]

SVMs F1 from 75.89 to 77.25

Conclusions

- Kernel methods and SVMs are useful tools to design language applications
- Kernel design still requires some level of expertise
- Engineering approaches to tree kernels
  - Basic Combinations
  - Canonical Mappings, e.g.
    - Node Marking
  - Merging of kernels in more complex kernels
- Easy modeling produces state-of-the-art accuracy in many tasks, RTE, SRL, QC, NER, RE
- SVM-Light-TK efficient tool to use them
Future (ongoing work)

- Once we have found the right kernel, are we satisfied?
- What about knowing the most relevant features?
- Can we speed up learning/classification at real-application scenario level?
- The answer is reverse kernel engineering:
  - Mine the most relevant fragments according to SVMs gradient
  - Use the linear space
- Software for reverse kernel engineering available in the next months

Thank you
References

References


References

References


References

References


An introductory book on SVMs, Kernel methods and Text Categorization
Non-exhaustive reference list from other authors


---

Non-exhaustive reference list from other authors

Non-exhaustive reference list from other authors


Non-exhaustive reference list from other authors

Non-exhaustive reference list from other authors


Non-exhaustive reference list from other authors

Author Index

Moschitti, Alessandro, 1