IIS at ImageCLEF 2015: Multi-label classification task

Antonio J Rodríguez-Sánchez[1], Sabrina Fontanella[1,2], Justus Piater[1], and Sandor Szedmak[1]

[1] Intelligent and Interactive Systems, Department of Computer Science, University of Innsbruck, Austria
{antonio.rodriguez-sanchez,justus.piater,sandor.szedmak}@uibk.ac.at
https://iis.uibk.ac.at/
[2] Department of Computer Science, University of Salerno, Italy
fontanellasabrina@gmail.com

Abstract. We propose an image decomposition technique that captures the structure of a scene. An image is decomposed into a matrix that represents the adjacency between the elements of the image and their distance. Images decomposed this way are then classified using a maximum margin regression (MMR) approach where the normal vector of the separating hyperplane maps the input feature vectors into the outputs vectors. Multiclass and multilabel classification are native to MMR, unlike other more classical maximum margin approaches, like SVM. We have tested our approach with the ImageCLEF 2015 multi-label classification task, obtaining high rankings at that task.

Keywords: ImageCLEF, Kronecker decomposition, Maximum Margin, MMR, SVM, multi-label classification, medical images

1 Introduction

Automatic image classification is a fundamental part of computer vision. An image is classified according to the visual content it contains. Tasks in image classification include if an image contains a certain object, person, animal or plant; if the image is from a street or it is indoors; or in the case that applies to this paper, if it is a medical figure and the type of medical images and/or graphs it contains. Image classification spans several decades from the first character or digit recognition challenges (still used today), such as the MNIST dataset [1] to more recent, challenging image classification tasks, such as the Pascal [2], Imagenet [3] or ImageCLEF [4, 5] challenges.

Image classification algorithms usually consist of a first step where keypoints or regions are found, which are then assigned a representation in terms of a feature vector. During training, the feature vectors extracted from a set of training images are usually grouped into histograms that approximate the distribution of the features for the different types of images. These histograms compose the input to a classification algorithm, such as an SVM (discriminative) or Naive Bayes (generative).

One of the central problems in exploring the general structure of an image is to recognize the relations between the objects appearing on the image. The task is not really the recognition of the objects but rather building a model on the structure: what belongs to what and how they can be related. It might be similar to the way how an animal could observe the world without labels attached to the object but only relying on relations among them in a certain environment. Those relations could provide the knowledge needed to identify scenes.

One of the most popular streams of machine learning research is to find efficient methods for learning structured outputs. Several researchers introduced similar approaches to these kind of problems [6–10]. Those methods directly incorporate the structural learning into a specially chosen optimization framework.

It is generally assumed that to learn a discriminating function when the output space is a labeled hierarchy is a much more complex problem than binary classification. In this paper we show that the complexity of this kind of problem can be detached from the optimization model and can be expressed by an embedding into Hilbert space. This allows us to apply a universal optimization model, processing inputs and outputs represented in a properly chosen Hilbert space which can solve the corresponding optimization task without tackling with the underlying structural complexity. The optimization model is an implementation of a certain type of maximum margin regression, an algebraic generalization of the well-known Support Vector Machine. The computational complexity of the optimization scales only with the number of input-output pairs and it is independent from the dimensions of both spaces. Furthermore its overall complexity is equal to a binary classification. Our approach can be easily extended towards other structural learning problems without giving up efficiency on the basic optimization framework.

Three fundamental steps are needed for structural learning:

Embedding The structures of the input and output objects are represented as abstract vectors in properly chosen Hilbert spaces reflecting the similarity and the dissimilarity of the objects.

Optimization The optimization phase is implemented via a universal solver which tries to find the best similarity based matching between the input and the output representations. Since these representation are expressed as general vectors, the optimizer needs not directly tackle the underlying structural complexity.

Inversion The optimizer provides a decision function which emits a vector. The inversion phase has to find the best fitting output structure by projecting the image vector back. This is often referred to as the pre-Image problem, see some alternatives presented in [10]. If the embedding is realized as a bijective mapping the inversion task is well defined.

We will make use of the following mathematical notation conventions in the rest of the paper: $\mathcal{X}$ stands for the space of the input objects, $\mathcal{Y}$ for the space of the outputs. $\mathcal{H}_\phi$ is a Hilbert space comprising the feature vectors, the images of the input vectors with respect to the embedding $\phi()$. $\mathcal{H}_\psi$ is a Hilbert space comprising the image of label vectors with respect to the embedding $\psi()$. $\mathbf{W}$ is a matrix representing the linear operator projecting the feature space $\mathcal{H}_\phi$ into $\mathcal{H}_\psi$. $\langle .,. \rangle_{\mathcal{H}_z}$ denotes the inner product in Hilbert space $\mathcal{H}_z$, $\|.\|_{\mathcal{H}_z}$ is the norm defined in Hilbert space $\mathcal{H}_z$. $\mathbf{tr}(\mathbf{W})$ is the trace of matrix $\mathbf{W}$. $\mathbf{dim}(\mathcal{H})$ is the dimension of the space $\mathcal{H}$. $\mathbf{x}_1 \otimes \mathbf{x}_2$ denotes the tensor product of the vectors $\mathbf{x}_1 \in \mathcal{H}_1$ and $\mathbf{x}_2 \in \mathcal{H}_2$, and it represents a linear operator $\mathbf{A} : \mathcal{H}_2 \rightarrow \mathcal{H}_1$ which acts on a vector $\mathbf{z} \in \mathcal{H}_2$ as $(\mathbf{x}_1 \otimes \mathbf{x}_2)\mathbf{z} \overset{\text{def}}{=} (\mathbf{x}_1\mathbf{x}_2')\mathbf{z} = \mathbf{x}_1\langle \mathbf{x}_2, \mathbf{z}\rangle_{H_2}$. $\langle \mathbf{A}, \mathbf{B}\rangle_F$ is the Frobenius inner product of a matrix represented by the linear operators $\mathbf{A}$ and $\mathbf{B}$ and it is defined by $\mathbf{tr}(\mathbf{A}'\mathbf{B})$. $\|\mathbf{A}\|_F$ stands for the Frobenius norm of a matrix represented by the linear operator $\mathbf{A}$ and defined by $\sqrt{\langle \mathbf{A}, \mathbf{A}\rangle_F}$. $\mathbf{A} \cdot \mathbf{B}$ is the element-wise(Schur) product of the matrices $\mathbf{A}$ and $\mathbf{B}$. $\mathbf{A}'$, $\mathbf{a}'$ is the transpose of any matrix $\mathbf{A}$ or any vector $\mathbf{a}$.

2 Image feature generation via decomposition

Let us consider a real 2D image decomposition, where we can expect that the points close to each other within continuous 2D blocks relate more strongly to each other than only considering their connection in 1D rows and columns. To represent the image decomposition, the Kronecker product is applied, which can be expressed as

$$\begin{array}{l} \mathbf{X} = \mathbf{A} \otimes \mathbf{B} \begin{bmatrix} A_{1,1}\mathbf{B} & A_{1,2}\mathbf{B} & \cdots & A_{1,n_A}\mathbf{B} \\ A_{2,1}\mathbf{B} & A_{2,2}\mathbf{B} & \cdots & A_{2,n_A}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m_A,1}\mathbf{B} & A_{m_A,2}\mathbf{B} & \cdots & A_{m_A,n_A}\mathbf{B} \end{bmatrix} \\ \mathbf{A} \in \mathbb{R}^{m_A \times n_A}, \mathbf{B} \in \mathbb{R}^{m_B \times n_B},\ m_X = m_A \times m_B, n_X = n_A \times n_B \end{array} \quad (1)$$

In the Kronecker decomposition the second component ($\mathbf{B}$) can be interpreted as a 2D filter of the image represented by the matrix $\mathbf{X}$. We can try to find a sequence of filters by the following procedure:

```
1. $k = 1$
2. $\mathbf{X}^{(k)} = \mathbf{X}$
3. DO
4. $d(A^{(k)}, B^{(k)}) = \min_{\mathbf{A}_k, \mathbf{B}_k} \|\mathbf{X}^{(k)} - \mathbf{A}^{(k)} \otimes \mathbf{B}^{(k)}\|^2$
5. IF $d(A^{(k)}, B^{(k)}) \leq \epsilon$ STOP
6. $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} - \mathbf{A}^{(k)} \otimes \mathbf{B}^{(k)}$
7. $k = k + 1$
8. Goto 3
```

The question is, if $\mathbf{X}$ is given, how do we compute $\mathbf{A}$ and $\mathbf{B}$? It turns out that the Kronecker decomposition can be carried out by Singular Value Decomposition (SVD) working on a reordered representation of the matrix $\mathbf{X}$.

For an arbitrary matrix $\mathbf{X}$ with size $m \times n$ the SVD is given by $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{U} \in \mathbb{R}^{m \times m}$ is an orthogonal matrix, $\mathbf{U}\mathbf{U}^T = \mathbf{I}_m$, of left singular vectors,

$\mathbf{V} \in \mathbb{R}^{n \times n}$, is an orthogonal matrix, $\mathbf{V}\mathbf{V}^T = \mathbf{I}_n$, of right singular vectors, and $\mathbf{S} \in \mathbb{R}^{m \times n}$, is a diagonal matrix containing the singular values with nonnegative components in its diagonal.

2.1 Reordering of the matrix

Since the algorithm solving the SVD problem does not depend directly on the order of the elements of the matrix [11], any permutation of the indexes, i.e. reordering the columns and(or) rows, preserves the same solution.

2.2 Kronecker decomposition as SVD

The solution to the Kronecker decomposition via the SVD can be found in [11]. This approach considers the aforementioned observation regarding the invariance of the SVD on the reordering of the matrix elements.

In order to show how the reordering of matrix $\mathbf{X}$ can help to solve the Kronecker decomposition problem we present the following example. The matrices in the Kronecker product

$$\begin{array}{ccc}
\mathbf{X} & & = \mathbf{A} \otimes \mathbf{B} \\
\left[\begin{array}{cc|cc|cc}
x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} \\
x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} \\
\hline
x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} \\
x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} \\
\hline
x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} \\
x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & x_{66}
\end{array}\right]
& = &
\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \otimes \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix},
\end{array}$$

can be reordered into

$$\begin{aligned}
\tilde{\mathbf{X}} = \tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}} &= \begin{bmatrix}
x_{11} & x_{13} & x_{15} & x_{31} & x_{33} & x_{35} & x_{51} & x_{53} & x_{55} \\
x_{12} & x_{14} & x_{16} & x_{32} & x_{34} & x_{36} & x_{52} & x_{54} & x_{56} \\
x_{21} & x_{23} & x_{25} & x_{41} & x_{43} & x_{45} & x_{61} & x_{63} & x_{65} \\
x_{22} & x_{24} & x_{26} & x_{42} & x_{44} & x_{46} & x_{62} & x_{64} & x_{66}
\end{bmatrix} \\
&= \begin{bmatrix} b_{11} \\ b_{12} \\ b_{21} \\ b_{22} \end{bmatrix} \otimes \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{21} & a_{22} & a_{23} & a_{31} & a_{32} & a_{33} \end{bmatrix},
\end{aligned}$$

where the blocks of $\mathbf{X}$ and the matrices $\mathbf{A}$ and $\mathbf{B}$ are vectorized in row wise order. In this vectorization we follow that order which is applied in most of the well known programming languages, C, Java, Python, MATLAB, instead of the column wise order, e.g. used in the Fortran language.

We can recognize that $\tilde{\mathbf{X}} = \tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}}$ can be interpreted as the first step in the SVD algorithm where we might apply the substitution $\sqrt{s}\mathbf{u} = \tilde{\mathbf{A}}$ and $\sqrt{s}\mathbf{v} = \tilde{\mathbf{B}}$. The proof that this reordering generally provides the correct solution to the Kronecker decomposition can be found in [11].

We can summarize the main steps of the Kronecker decomposition in the following steps:

1. Reorder(reshape) the matrix,
2. Compute the SVD decomposition,
3. Compute the approximation of $\tilde{\mathbf{X}}$ by $\tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}}$
4. Invert the reordering.

This kind of Kronecker decomposition is often referred as Nearest Orthogonal Kronecker Product as well [11].

3 Learning task

The learning task that we are going to solve is the following: There is a set - called sample - of pairs of output and input objects $\{(y_i, x_i) : y_i \in \mathcal{Y},\ x_i \in \mathcal{X},\ i = 1, \ldots, m,\}$ independently and identically chosen out of an unknown multivariate distribution $\mathcal{P}(Y, X)$. Here we would like to emphasize that the input and the output objects can be arbitrary, e.g. they may be graphs, matrices, functions, probability distributions etc. To these objects, let's consider two functions $\phi : \mathcal{X} \rightarrow \mathcal{H}_\phi$ and $\psi : \mathcal{Y} \rightarrow \mathcal{H}_\psi$ mapping the input and output objects respectively into linear vector spaces, called from now on, the feature space in case of the inputs and the label space when the outputs are considered.

The objective is to find a linear function acting on the feature space

$$f(\boldsymbol{\phi}(x)) = \mathbf{W}\boldsymbol{\phi}(x) + \mathbf{b}, \tag{2}$$

that produces a prediction of every input object in the label space and in this way could implicitly give back a corresponding output object. Formally we have

$$y = \psi^{-1}(\psi(y)) = \psi^{-1}(f(\boldsymbol{\phi}(x))). \tag{3}$$

The learning procedure can be summarized as follows:

$$\begin{array}{ll}
\text{Embedding} & \begin{array}{l} \phi : \overbrace{\text{input space}}^{\mathcal{X}} \rightarrow \overbrace{\text{feature space}}^{\mathcal{H}_\phi}, \\ : \overbrace{\text{output space}}^{\mathcal{Y}} \rightarrow \overbrace{\text{label space}}^{\mathcal{H}_\psi}, \end{array} \\
\begin{array}{l}\text{Similarity} \\ \text{transformation}\end{array} & \widetilde{\mathbf{W}} = (\mathbf{W}, \mathbf{b}) \Rightarrow \boldsymbol{\psi}(y) \sim \widetilde{\mathbf{W}}\boldsymbol{\phi}(x), \\
\text{Inversion} & \boldsymbol{\psi}^{-1} : \overbrace{\text{label space}}^{\mathcal{H}_\psi} \rightarrow \overbrace{\text{output space}}^{\mathcal{Y}}.
\end{array}$$

4 Optimization model

4.1 The "Classical" scheme of Support Vector Machine (SVM)

In the framework of the Support Vector Machine the outputs represent two classes and the labels are chosen out of the set $y_i \in \{-1, +1\}$. The aim is to find a separating hyperplane, via its normal vector, such that the distance between the elements of the two classes, called margin, is the largest one measured in the direction of this normal vector. This base scheme can be extended allowing some sample items to fall closer to the separating hyperplane than to the margin.

This learning scenario can be formulated as an optimization problem:

$$\begin{array}{ll} \min & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\mathbf{1}'\boldsymbol{\xi} \\ \text{w.r.t.} & \mathbf{w} : \mathcal{H}_\phi \rightarrow \mathbb{R},\ \text{normal vector} \\ & b \in \mathbb{R},\ \text{bias},\ \boldsymbol{\xi} \in \mathbb{R}^m,\ \text{error vector} \\ \text{s.t.} & y_i(\mathbf{w}'\boldsymbol{\phi}(\mathbf{x}_i) + b) \geq 1 - \xi_i \\ & \boldsymbol{\xi} \geq \mathbf{0},\ i = 1, \ldots, m. \end{array}$$

4.2 Reinterpretation of the normal vector w

The normal vector $\mathbf{w}$ formally behaves as a linear transformation acting on the feature vectors whose capabilities can be even further extended. This extension can be characterized briefly in the following way

SVM	**ExtendedView**
– $\mathbf{w}$ is the normal vector of the separating hyperplane.	– $\mathbf{W}$ is a linear operator projecting the feature space into the label space.
– $y_i \in \{-1, +1\}$ binary outputs.	– $y_i \in \mathcal{Y}$ arbitrary outputs
– The labels are equal to the binary objects.	– $\boldsymbol{\psi}(y_i) \in \mathcal{H}_\psi$ are the labels, the embedded outputs in a linear vector space

If we apply a one-dimensional normalized label space invoking binary labels $\{-1, +1\}$ in the general framework, one can restore the original scenario of the SVM, and the normal vector is a projection into the one dimensional label space.

To summarize the learning task, we end up in the following optimization problem when compared to the original primal form of the SVM:

Primal problems for maximum margin learning

	Binary class learning Support Vector Machine(SVM)	Vector label learning Maximum Margin Regression(MMR)
min	$\frac{1}{2}\boxed{\|\mathbf{w}\|_2^2} + C\mathbf{1}'\boldsymbol{\xi}$	$\frac{1}{2}\boxed{\|\mathbf{W}\|_F^2} + C\mathbf{1}'\boldsymbol{\xi}$
w.r.t.	$\boxed{\mathbf{w} : \mathcal{H}_\phi \rightarrow \mathbb{R},}$ normal vector	$\boxed{\mathbf{W} : \mathcal{H}_\phi \rightarrow \mathcal{H}_\psi,}$ linear operator,
	$\boxed{b \in \mathbb{R},}$ bias,	$\boxed{\mathbf{b} \in \mathcal{H}_\psi,}$ translation(bias),
	$\boldsymbol{\xi} \in \mathbb{R}^m$, error vector	,
s.t.	$\boxed{y_i(\mathbf{w}'\boldsymbol{\phi}(\mathbf{x}_i) + b)} \geq 1 - \xi_i,$	$\boxed{\left\langle \boldsymbol{\psi}(\mathbf{y}_i), \mathbf{W}\boldsymbol{\phi}(\mathbf{x}_i) + \mathbf{b} \right\rangle_{\mathcal{H}_\psi}} \geq 1 - \xi_i,$
	$\boldsymbol{\xi} \geq \mathbf{0},\ i = 1, \ldots, m.$	

In the extended formulation we exploit the fact that the Frobenius norm and inner product correspond to the linear vector space of matrices with dimension equal to the number of elements of the matrices, hence it gives an isomorphism between the space spanned by the normal vector of the hyperplane occurring in the SVM and the space spanned by the linear transformations.

One can recognize that if no bias term is included in the MMR problem then we have a completely symmetric relationship between the label and the feature space via the representations of the input and the output items, namely

$$\left\langle \boldsymbol{\psi}(\mathbf{y}_i), \mathbf{W}\boldsymbol{\phi}(\mathbf{x}_i) \right\rangle_{\mathcal{H}_\psi} = \left\langle \mathbf{W}^*\boldsymbol{\psi}(\mathbf{y}_i), \boldsymbol{\phi}(\mathbf{x}_i) \right\rangle_{\mathcal{H}_\phi} = \left\langle \boldsymbol{\phi}(\mathbf{x}_i), \mathbf{W}^*\boldsymbol{\psi}(\mathbf{y}_i) \right\rangle_{\mathcal{H}_\phi}.$$

Thus, in predicting the input items as the image of a linear function defined on the outputs the, adjugate of $\mathbf{W}$, $\mathbf{W}^*$ is involved. This adjugate is equal to the transpose of the matrix representation of $\mathbf{W}$ whenever both, the label space and the feature space are finite dimensional spaces.

4.3 Dual problem

The dual problem of MMR presented in (4.2) is given by

$$\begin{array}{ll} \min & \sum_{i,j=1}^m \alpha_i\alpha_j \overbrace{\langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)\rangle}^{\kappa_{ij}^{\phi}} \overbrace{\langle \boldsymbol{\psi}(\mathbf{y}_i), \boldsymbol{\psi}(\mathbf{y}_j)\rangle}^{\kappa_{ij}^{\psi}} - \sum_{i=1}^m \alpha_i, \\ \text{w.r.t.} & \alpha_i \in \mathbb{R}, \\ \text{s.t.} & \sum_{i=1}^m (\boldsymbol{\psi}(\mathbf{y}_i))_t \alpha_i = 0, \ t = 1, \dots, \dim(\mathcal{H}_\psi), \\ & 0 \le \alpha_i \le C, \ i = 1, \dots, m, \end{array}$$

where κ_{ij}^{ϕ} kernel items correspond to the feature vectors, and κ_{ij}^{ψ} kernel items correspond to the label vectors.

The symmetry of the objective function is clearly recognizable showing that the underlying problem without bias is completely reversible. The explicit occurrences of the label vectors can be transformed into implicit ones by exploiting that the feasibility domain covered by the constraints: $\sum_{i=1}^m (\boldsymbol{\psi}(\mathbf{y}_i))_t \alpha_i = 0, \ t = 1, \dots, \dim(\mathcal{H}_\psi)$, coincides with the domain of $\sum_{i=1}^m \kappa_{ij}^{\psi}\alpha_i = 0, \ j = 1, \dots, m$ involving only inner products of the label vectors.

In the case of the original SVM κ_{ij}^{ψ} collapses into the product $y_i y_j$ of the binary labels $+1$ and -1.

4.4 Prediction

After solving the dual problem with the help of the optimum dual variables we can write up the optimal linear operator

$$\mathbf{W} = \sum_{i=1}^m \alpha_i \boldsymbol{\psi}(\mathbf{y}_i)\boldsymbol{\phi}(\mathbf{x}_i)'.$$

We can solve this expression by comparing it to the corresponding formula which gives the optimal solution to the SVM, i.e. $\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i)$. The new part includes the vectors representing the output items which in the SVM were only scalar values but we could say in the new interpretation that they are one-dimensional vectors. With the expression of the linear operator $\mathbf{W}$ at hand, the prediction to a new input item $\mathbf{x}$ can be written as

$$\boldsymbol{\psi}(\mathbf{y}) = \mathbf{W}\boldsymbol{\phi}(\mathbf{x}) = \sum_{i=1}^m \alpha_i \boldsymbol{\psi}(\mathbf{y}_i) \boxed{\underbrace{\langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x})\rangle}_{\kappa^{\phi}(\mathbf{x}_i, \mathbf{x})}}.$$

which involves only the input kernel κ^{ϕ} and provides the implicit representation of the prediction $\boldsymbol{\psi}(\mathbf{y})$ to the corresponding output $\mathbf{y}$.

Because only the implicit image of the output is given, we need to invert the function ψ to obtain its corresponding $\mathbf{y}$. This inversion problem is called the pre-image problem. Unfortunately there is no general procedure to do that. We mention here a scheme that can be applied when the set of all possible outputs is finite with a reasonable small cardinality. The meaning of the "reasonable small" cardinality depends on the given problem, e.g. how expensive is to compute the inner product between the output items in the label space where they are represented.

At the conditions mentioned we can follow this scenario

$$\begin{aligned}
\mathbf{y}^* &= \arg\max_{\mathbf{y}\in\widetilde{\mathcal{Y}}} \boldsymbol{\psi}(\mathbf{y})'\mathbf{W}\boldsymbol{\phi}(\mathbf{x}) \\
&= \arg\max_{\mathbf{y}\in\widetilde{\mathcal{Y}}} \sum_{i=1}^{m} \alpha_i \overbrace{\langle \boldsymbol{\psi}(\mathbf{y}), \boldsymbol{\psi}(\mathbf{y}_i)\rangle}^{\kappa^{\psi}(\mathbf{y},\mathbf{y}_i)} \overbrace{\langle \boldsymbol{\phi}(\mathbf{x}_i)'\boldsymbol{\phi}(\mathbf{x})\rangle}^{\kappa^{\phi}(\mathbf{x}_i,\mathbf{x})} \\
&\quad \text{where } \mathbf{y} \in \widetilde{\mathcal{Y}} = \{\mathbf{y}_1, \dots, \mathbf{y}_N\} \Leftarrow \text{is the set of the possible outputs}
\end{aligned}$$

The main advantage of this approach is that it requires only the inner products in label space. in addition to this, it is independent from the representation of the output items and can be applied in any complex structural learning problem, e.g. on graphs. Probably the best candidate for $\widetilde{\mathcal{Y}}$ could be the training set.

4.5 Hierarchy learning

As mentioned above in this paper we focus on the case where the output space is a labeled hierarchy (Figure 1a). The hierarchy learning is realized via an embedding of each path going from a node to the root of the tree. Let V be the set of nodes in the tree. A path $p(v) \subset V$ is defined as a shortest path from the node v to the root of the tree and its length is equal to $|p(v)|$. The set $I = 1, \dots, |V|$ gives an indexing of the nodes. The embedding is realized by a vector valued function $\boldsymbol{\psi} : V \rightarrow \mathbb{R}^{|V|}$, and the components of $\boldsymbol{\psi}(v)$ are given by

$$\psi(v)_i = \begin{cases} r & \text{if } v_i \notin p(v), \\ sq^k & \text{if } v_i \in v(p) \text{ and } k = |p(v)| - |p(v_i)|, \end{cases} \tag{4}$$

where r, q, s are the parameters of the embedding. The parameter q expresses the diminishing weight of the nodes being closer to the root. If $q = 0$, assuming $0^0 = 1$, then the intermediate nodes and the root are disregarded, thus we have a simple multiclass classification problem. The value of r can be 0 but some experiments show it may help to improve the classification performance. We might conjecture the best choice of the parameters are those which minimize the correlation between all pairs of the label vectors.

4.6 Input and output kernels

For the concrete learning task we need to construct the input and output kernels. To build the input kernel, the second component of the Kronecker decomposition of each image - the matrix **B** in (1) - is used. The inner product between those matrices is computed by applying the Frobenius inner product. The output kernel is created from the inner products of the vectors representing the path in the hierarchy in (4).

5 Experimental evaluation

5.1 ImageCLEF multi-label classification [4, 5]

The challenge we participated was the characterization of compound figures. These figures contain subfigures from different types and sources (see figure 1b,c for two examples). The task consists of labeling the compound figures with each of the 30 classes that appear in the hierarchy in figure 1a without knowing where the separation lines are. The training set consists of 1,071 figures, the test set consists of 927 figures.

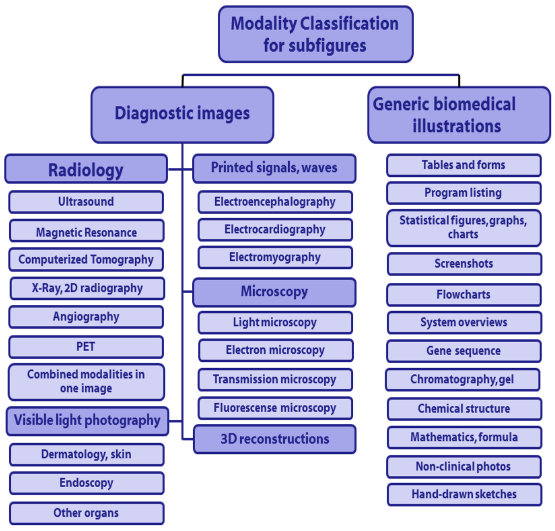
5.2 Results on the challenge

In the computation of the prediction results, a 5-fold cross-validation procedure is applied. The original dataset is split uniformly and randomly into 5 equal parts. Then, each part is chosen as test data in a loop and the remaining four parts are taken as training. In the learning procedure, first, a kernel is computed from the corresponding features. Parameters corresponding to each kernel are found by cross validation restricted to the training data, namely it is divided into validation test and validation training parts. Then the learner is trained only on the validation training items. The values of the parameters are chosen which maximize the $F1$ score on the validation test. We will report here on two types of kernels: polynomial and Gaussian.

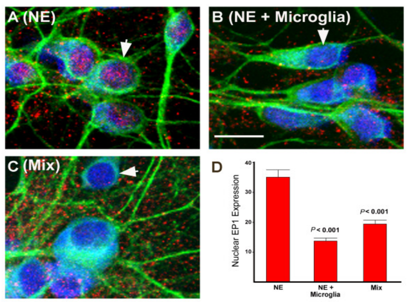
We submitted ten runs to the challenge providing our predictions on the labels for the test set (before it was made public). For the ten runs, we used a third degree polynomial kernel, the only factor changing at each run was the random selection in the 5-fold cross-validation. Generation of the feature vectors for the training set took around 60 minutes. The training of MMR would take around one minute, and obtaining the labels for the test set took less than a minute for each run. The ImageCLEF organizers provided the Hamming loss, which is a classical measure for multi-label classification tasks and evaluates the fraction of wrong labels to the total number of labels. The perfect case would be obtaining a Hamming loss of 0. Hamming loss values for the challenge in our case were exceptionally low (very close to 0) and ranged from 0.0671 to 0.0817.

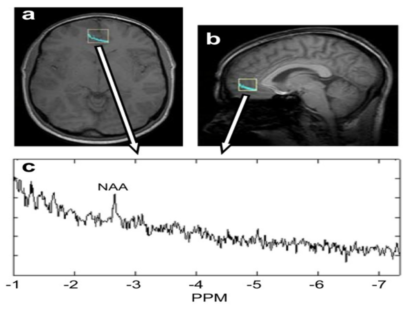
Once the test set was available we performed extra evaluation. We used three other different evaluation measures that are popular in multi-label classification, namely precision, recall and their combination into the F1 score. They are given by a combination of the true positives T_p, false positives F_p and false negatives F_n:

$$P = \frac{T_p}{T_p+F_p},\ R = \frac{T_p}{T_p+F_n},\ F1 = \frac{2PR}{P+R} \tag{5}$$



(a)





(b) (c)

Fig. 1. a) The hierarchy of classes in the ImageCLEF 2015 [4, 5] multi-label challenge; b) and c) are two figure examples.

where P is the precision and R is the recall. Here, the perfect case would have a recall value of 1 for any precision. The $F1$ measure combines both values into one so that false positives and false negatives are taken into account in this one value. Precision-Recall curves for six different Kronecker 2D filter sizes (4, 8, 12, 20, 28, 34) are given in figure 2a for polynomial kernerls of different degrees and in figure 3a for Gaussian kernels having different standard deviations. Their respective F1 scores are in figures 2b and 3b. The parameter for the Precision-Recall curve (and the F1 plot) in the polynomial kernel was the degree of the polynomial, from 1 to 10. The parameter that was varied in the Gaussian kernel to generate its Precision-Recall curve (and the F1 plot) was the standard deviation of the Gaussian: 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 2, 5 and 10.

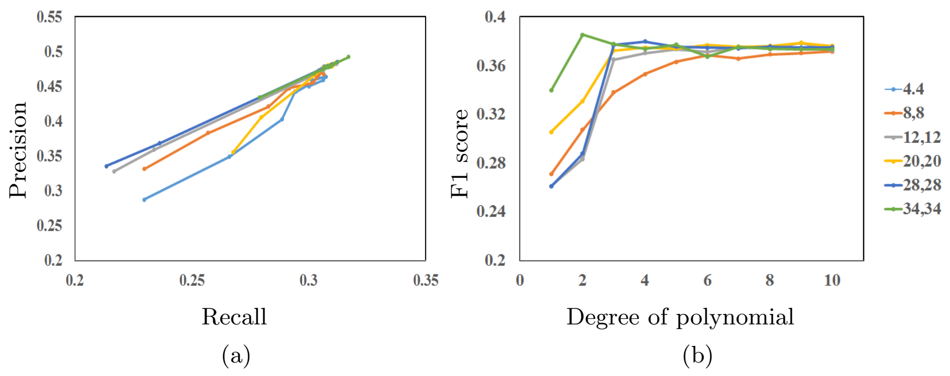


Fig. 2. Results for six filter sizes: 4, 8, 12, 20, 18 and 32, training with a polynomial kernel of degrees 1 to 10. a) Precision and Recall, b) F1 score.

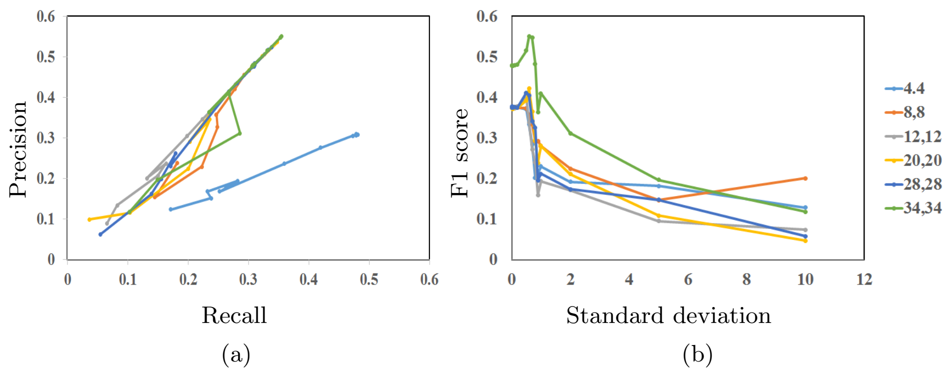


Fig. 3. Results for six filter sizes: 4, 8, 12, 20, 18 and 32, training with a Gaussian kernel with *stdev* 0.01 to 10. a) Precision and Recall, b) F1 score.

These results show that larger filter sizes provide better results, although at the largest filter sizes, Precision, Recall and F1 score are very similar. Regarding kernels, when using a polynomial kernel, there is a dramatically increase in F1 scores when using a cubic kernel as compared to a linear or quadratic one. Although at kernels of degree 4 and larger, F1 scores are very similar. In the case of a Gaussian kernel, the best scores happen at standard deviations smaller than 1, although values in the middle (e.g. 0.5, 0.6) provide better results than very small values (e.g. 0.01, 0.05). The best F1 score using a polynomial kernel was 0.38, in the case of a Gaussian kernel, the highest F1 score was 0.43.

6 Conclusions

We have presented an approach based on a structured decomposition of the environment. For example, elements appearing on a scene can be incorporated into a graph in which the objects play the role of the vertices and the edges related to the distances between those objects. Then the knowledge about the environment can be represented by the adjacency matrix of the graph. By decomposing the image matrix into a similar structure, e.g. into a sequence of Kronecker products, the structure behind the scene could be captured. For classification we have applied a version of a maximum margin based regression (MMR) technique [12]. MMR relies on the fact that the normal vector of the separating hyperplane can be interpreted as a linear operator mapping the feature vectors of input items into the space of the feature vectors of the outputs. The evaluation of our methodology in the ImageCLEF 2015 [4, 5] multi-label challenge provided promising results.

7 Acknowledgement

The research leading to these results received funding from the EU 7th Framework Programme FP7/2007-2013 under grant agreement no. 270273, Xperience.

References

1. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324
2. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision **111**(1) (2014) 98–136
3. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) (2015)
4. Villegas, M., Müller, H., Gilbert, A., Piras, L., Wang, J., Mikolajczyk, K., de Herrera, A.G.S., Bromuri, S., Amin, M.A., Mohammed, M.K., Acar, B., Uskudarli, S., Marvasti, N.B., Aldana, J.F., del Mar Roldán García, M.: General Overview of ImageCLEF at the CLEF 2015 Labs. Lecture Notes in Computer Science. (2015)
5. García Seco de Herrera, A., Müller, H., Bromuri, S.: Overview of the ImageCLEF 2015 medical classification task. In: Working Notes of CLEF 2015 (Cross Language Evaluation Forum). CEUR Workshop Proceedings (September 2015)
6. Taskar, B., Guestrin, C., Koller, D.: Max-margin markov networks. In: NIPS 2003. (2003)
7. Altun, Y., Tsochantaridis, I., Hofmann, T.: Hidden markov support vector machines. In: ICML'03. (2003) 3–10
8. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research (JMLR) **6(Sep)** (2005) 1453–1484
9. Rousu, J., Saunders, C., Szedmak, S., Shawe-Taylor, J.: Learning hierarchical multi-category text classification models. In: ICML. (2005)
10. Bakir, G., Hofman, T., B. Schölkopf, A.J.S., Taskar, B., Vishwanathan, S.V.N., eds.: Predicting Structured Data. MIT Press (2007)
11. Loan, C.: The ubiquitous kronecker product. Journal of Computational and Applied Mathematics **123** (2000) 85–100 The nearest Kronecker product.
12. Xiong, H., Szedmak, S., Piater, J.: Scalable, Accurate Image Annotation with Joint SVMs and Output Kernels. Neurocomputing (2015)