Support Vector Machines

a practical guide

Danilo Croce

Support Vector Machines

- Consideriamo un problema di classificazione binaria, a partire da uno spazio di input $X \subseteq \Re^n$ e uno spazio di output $Y = \{-1, 1\}$
- Training set S: $S = \{(x_1, y_1),. . . (x_l, y_l)\} \in (X \times Y)$
- Vogliamo apprendere una relazione che leghi X a Y
- Nel caso di classificazione questo si traduce in trovare una regola di decisione f che, dato un punto x dello spazio di input X, gli associ l'etichetta −1 o 1, a seconda della classe di appartenenza.

Support Vector Machines

- Un classificatore binario viene modellato a partire da una funzione
- $f : X \subseteq \Re^n \rightarrow \Re$:
  - if $f(x) \geq 0$ +1
  - if $f(x) < 0$ - 1
- La funzione di classificasione è : sign(f(x))

Support Vector Machines

The overweight's Example

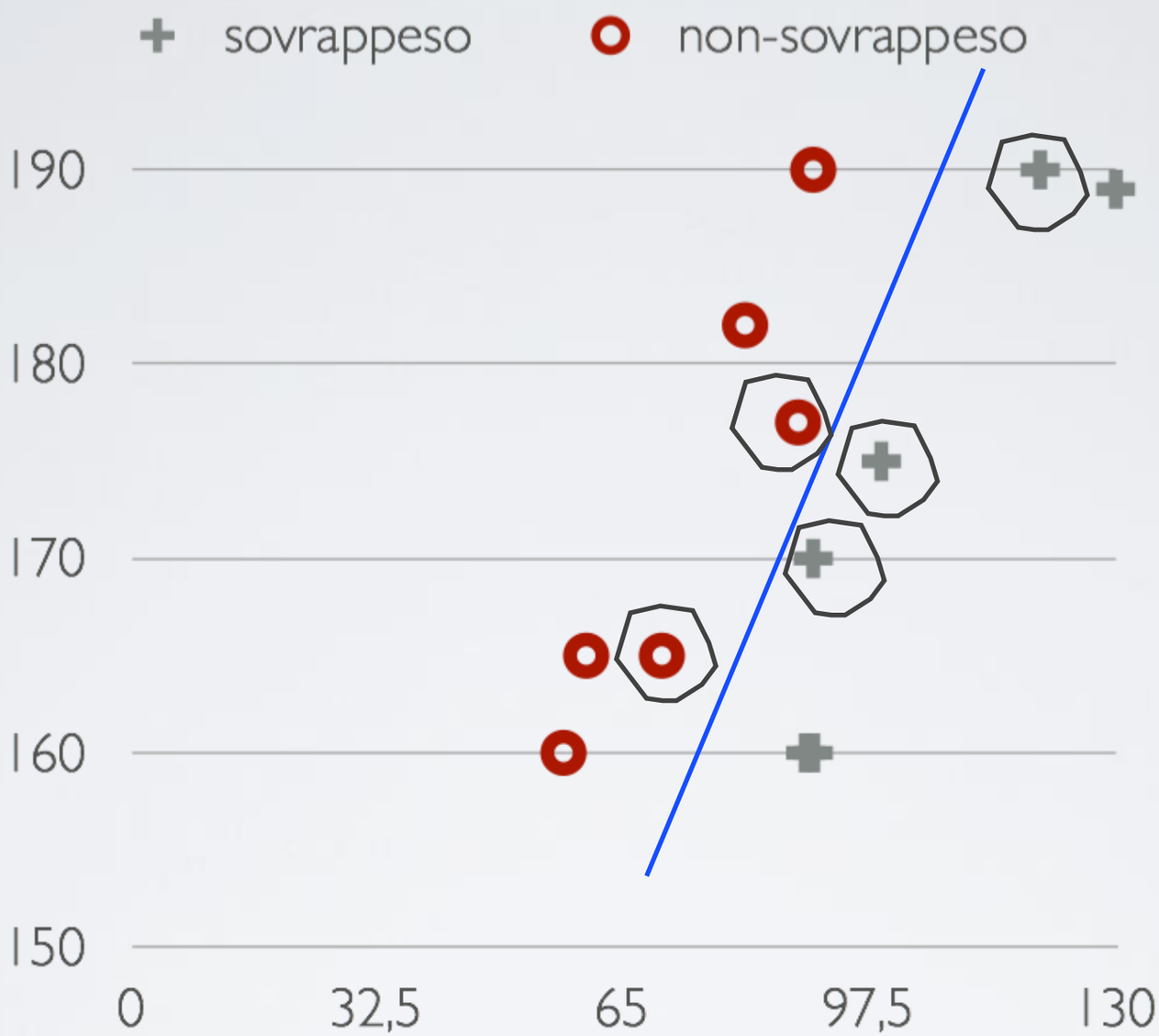
- Esempio
  - Consideriamo il seguente problema:
  - Vogliamo apprendere la funzione che definisce l'appartenenza o meno di un individuo alla classe dei sovrappeso
  - Usando un approccio orientato ai dati, è possibile collezionare esempi per sfruttare informazioni legate alla "forma" di una persona

The overweight's Example

Features extraction

(x_1,x_2,x_3,x_4)	y
1) Gender: man, Years: 47, Weigh:130, Height:189	**Sovrappeso**
2) Gender: man, Years: 23, Weigh: 88, Height: 177	**Sovrappeso**
3) Gender: man, Years: 16, Weigh: 52, Height: 160	**Sovrappeso**
5) Gender: man, Years: 65, Weigh:120, Height: 190	**Sovrappeso**

height



weigh

Support Vectors

$$f(x) = \text{sign}(w\ x + b)$$

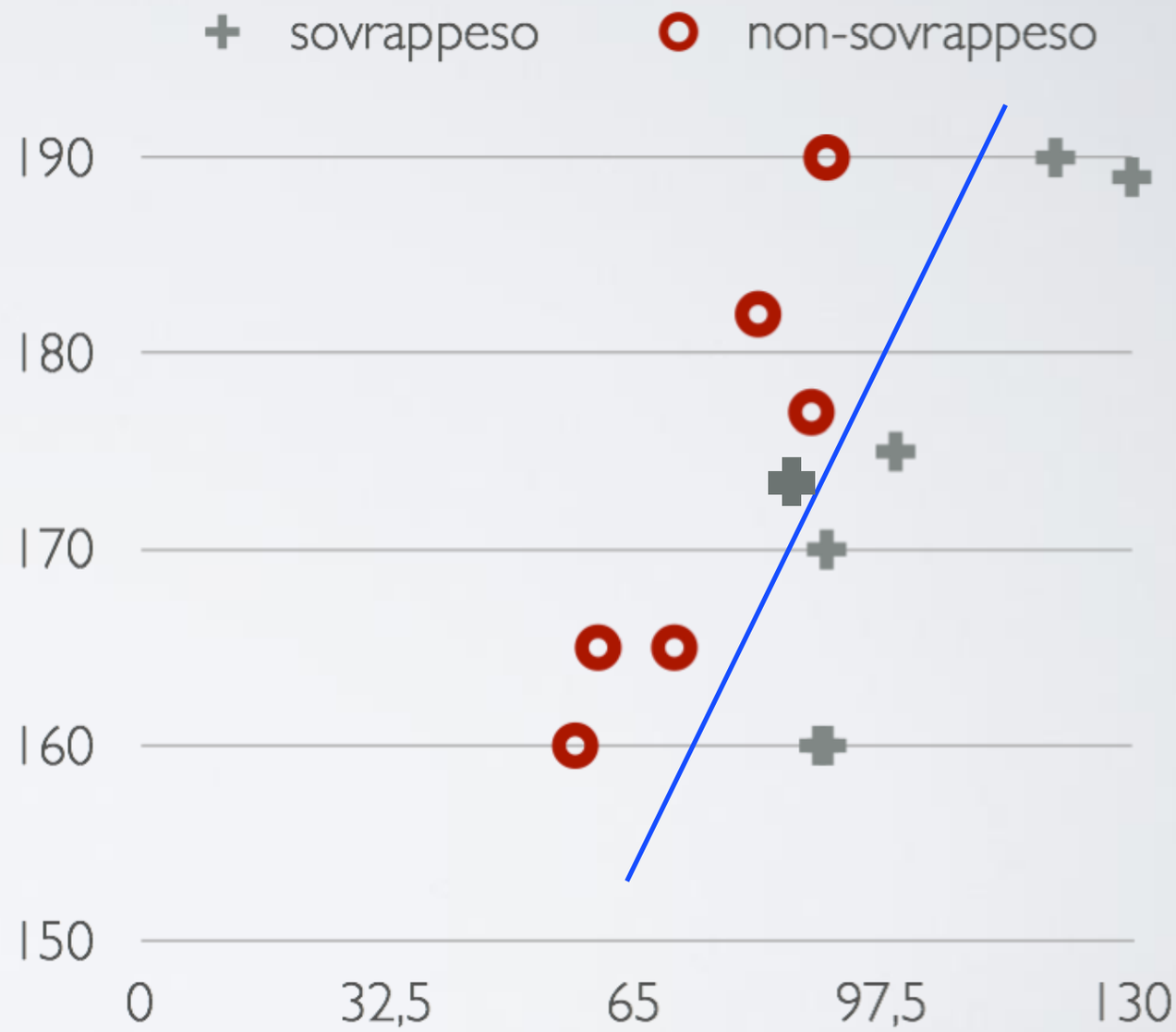
Svm Optimization
trade-off between training error and margin

The soft-margin SVM allows classification errors in the training set

- The trade-off parameter C need to appropriately chosen
- A very high value of C corresponds to the hard margin SVM

Minimize: $\frac{1}{2}||\vec{w}||^2 + C\sum_k \xi_k$

subject to: $\forall k : y_k[\vec{w} \cdot \vec{x_k} + b] \geq 1 - \xi_k$



SVM OPTIMIZATION
Adjusting the cost of false positives vs. false negatives

- As we can deal with unbalanced numbers of positive and negative examples, two different cost factors $\mathbf{C_+}$ and $\mathbf{C_-}$ are employed.

$$\begin{array}{ll} \text{Minimize:} & \frac{1}{2}||\vec{w}||^2 + C_+ \sum_{i:y_i=1} \xi_i + C_- \sum_{j:y_j=-1} \xi_j \\ \text{subject to:} & \forall k : y_k[\vec{w} \cdot \vec{x_k} + b] \geq 1 - \xi_k \end{array}$$

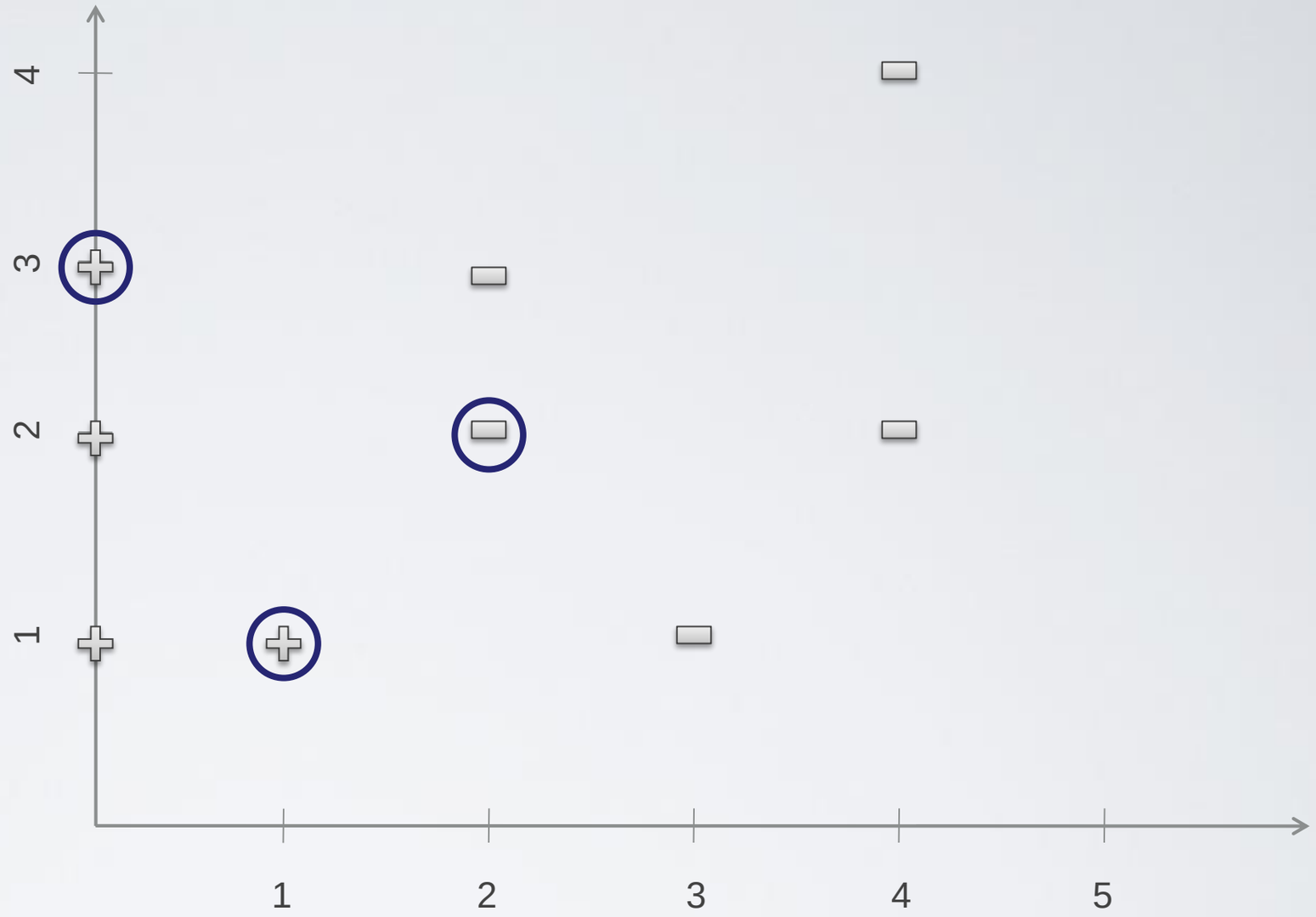
- They allow to adjust the cost of false positives vs. false negatives
- It is usually expressed by a unique parameter

$$J = C_+ / C_-$$

A pratical example

Positive Ex.
a1 (1,1)
a2 (0,1)
a3 (0,2)
a4 (0,3)
Negative Ex.
b1 (3,1)
b2 (2,2)
b3 (2,3)
b4 (4,2)
b5 (4,4)

SVM_LIGHT INPUT
+1 1:1 2:1
+1 2:1
+1 2:2
+1 2:3
-1 1:3 2:1
-1 1:2 2:2
-1 1:2 2:3
-1 1:4 2:2
-1 1:4 2:4



`svm_learn -j 1 -c 100 input model`

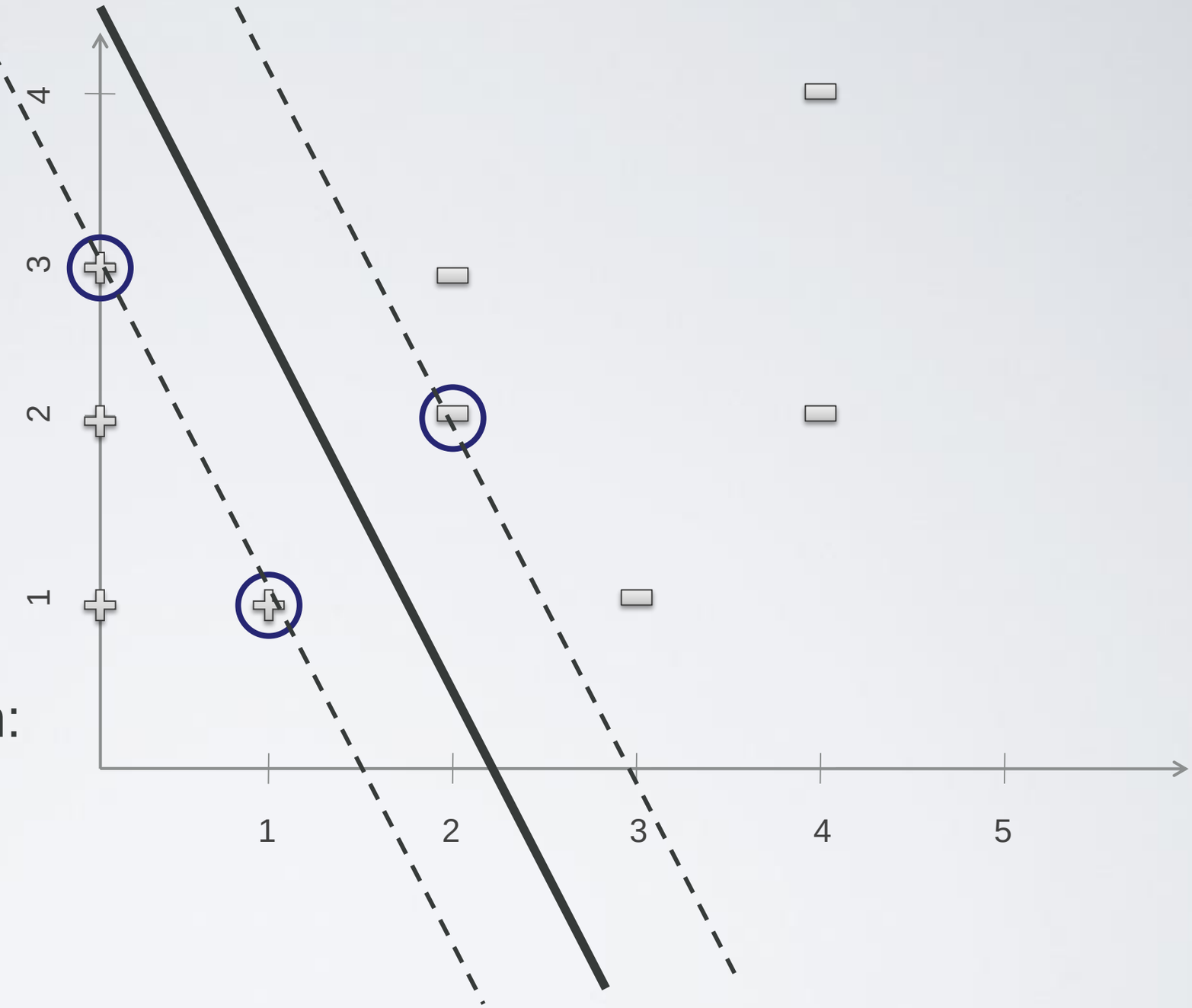
J=1 - c=100 : Hard Margin

SVM_LIGHT OUTPUT
-3.0004899 # threshold b
0.22219500340788700 39 2:3 #
0.88902498296056364 77 1:1 2:1 #
-1.1112199863684506 51 1:2 2:2 #

	α_i	COORD
sv_1	0.22219	(0,3)
sv_2	0.88902	(1,1)
sv_3	-1.11121	(2,2)
b	-3	

SIMPLE EXAMPLE

	α_i	COORD
sv_1	0.22219	(0,3)
sv_2	0.88902	(1,1)
sv_3	-1.111219	(2,2)
b	-3	



Hyperplane Equation:

$$w \cdot x - b = 0$$

$$w = \sum \alpha_i \cdot sv_i =$$

$$w = 0.2222 \cdot (0,3) + 0.8890 \cdot (1,1) - 1.1112 \cdot (2,2)$$

$$w = (-1.3334 , -0.6667)$$

$$-1.3334 \cdot x_1 - 0.6667 \cdot x_2 + 3 = 0$$

SVM Light

http://svmlight.joachims.org/

- Piattaforma che implementa l'algoritmo SVM
- Script per l'addestramento:
  - svm_learn [options] train_file model_file
- Script per la classificatione:
  - svm_classify [options] test_file model_file output_file

SVM Light
learning phase

svm_learn [options] train_file model_file
Available options are:

General options:
-? - this help
-v [0..3] - verbosity level (default 1)

Learning options:
-z {c,r,p} - select between classification (c), regression (r), and preference ranking (p) (see [Joachims, 2002c]) (default classification)
-c float - C: trade-off between training error and margin (default [avg. x*x]^-1)
-w [0..] - epsilon width of tube for regression (default 0.1)
-j float - Cost: cost-factor, by which training errors on positive examples outweight errors on negative examples (default 1) (see [Morik et al., 1999])

SVM Light
learning phase

```
<line> .=. <target> <feature>:<value> <feature>:<value>.. #info

<target> .=. {+1,-1}

<feature> .=. <integer>

<value> .=. <float>

<info> .=. <string>
```

weka VS svmlight input syntax

```
@relation segment

@attribute rawgreen-mean numeric
@attribute exred-mean numeric
@attribute exblue-mean numeric
@attribute exgreen-mean numeric
@attribute value-mean numeric
@attribute saturation-mean numeric
@attribute hue-mean numeric
@attribute class {brickface,sky,foliage,cement,window,path,grass}

@data
0,0,1,0.222222,6.22222,33.3185,29.0741,26.3333,35.2222,25.6667,-8.22222,18.4444,-10.2222,35.2222,0.271208,-2.04915,path
0,0,1.11111,0.607407,1.05556,0.462963,17.5185,13.1111,17.8889,21.5556,-13.2222,1.11111,12.1111,21.5556,0.393002,2.69011,grass
0,0,0.166667,0.077778,0.333333,0.088889,0.444444,0,1.33333,0,-1.33333,2.66667,-1.33333,1.33333,0.777778,-2.0944,foliage
0,0,3.05556,15.263,3.66667,6.08889,8.18519,6.55556,6.44444,11.5556,-4.88889,-5.22222,10.1111,11.5556,0.486717,2.09315,grass
0,0,0.055556,0.136083,0.111111,0.172133,1.25926,0.777778,3,0,-1.44444,5.22222,-3.77778,3,1,-1.82221,window
0,0,0.611111,0.240741,0.944445,0.32963,2.77778,0.444444,6.44444,1.44444,-7,11,-4,6.44444,0.938492,-2.26954,foliage
0,0,0.944444,1.06284,1.77778,1.31092,126.222,115.111,142.222,121.333,-33.3333,48,-14.6667,142.222,0.190625,-2.33375,sky
0,0,1.61111,0.742868,4.16667,2.12655,58,51.8889,72.4444,49.6667,-18.3333,43.3333,-25,72.4444,0.314281,-1.99159,path
```

Weka file: dichiarazione esplicita delle features, e delle classi da apprendere in multiclassificazione, la classe di un esempio è indicata a fine riga

```
+1  3:1 4:0.222222 5:6.22222 6:33.3185 7:29.0741 8:26.3333 9:35.2222 10:25.6667 11:-8.22222 12:18.44
-1  3:1.11111 4:0.607407 5:1.05556 6:0.462963 7:17.5185 8:13.1111 9:17.8889 10:21.5556 11:-13.2222 1
-1  3:0.166667 4:0.077778 5:0.333333 6:0.088889 7:0.444444 9:1.33333  11:-1.33333 12:2.66667 13:-1.33333 14:1.33333 15:0.777778 16:-2.0944
-1  3:3.05556 4:15.263 5:3.66667 6:6.08889 7:8.18519 8:6.55556 9:6.44444 10:11.5556 11:-4.88889 12:-5.22222 13:10.1111 14:11.5556 15:0.486717 16:2.093
-1  3:0.055556 4:0.136083 5:0.111111 6:0.172133 7:1.25926 8:0.777778 9:3  11:-1.44444 12:5.22222 13:-3.77778 14:3 15:1 16:-1.82221
-1  3:0.611111 4:0.240741 5:0.944445 6:0.32963 7:2.77778 8:0.444444 9:6.44444 10:1.44444 11:-7 12:11 13:-4 14:6.44444 15:0.938492 16:-2.26954
-1  3:0.944444 4:1.06284 5:1.77778 6:1.31092 7:126.222 8:115.111 9:142.222 10:121.333 11:-33.3333 12:48 13:-14.6667 14:142.222 15:0.190625 16:-2.33375
+1  1:0.1 3:1.611 4:0.742868 5:4.16667 6:2.12655 7:58 8:51.8889 9:72.4444 10:49.6667 11:-18.3333 12:43.3333 13:-25 14:72.4444 15:0.314281 16:-1.99159
```

smvlight file: classificazione binaria, le istanze positive sono indicate con +1 ad inizio riga, è indicato esplicitamente l'indice di feature

SVM Light
Sintassi esempi

- Le feauture devo essere in ordine crescente:
  - 1:1 3:1 2:1 no!!
  - 1:1 2:1 3:1 si!!
- In svm light è possibile omettere le feature che hanno valore nullo
  - 1:1 4:1 10:3 ....

SVM Light

learning phase

svm_learn train.dat file_model

```
Scanning examples...done
Reading examples into memory...OK. (2 examples read)
Optimizing.done. (2 iterations)
Optimization finished (0 misclassified, maxdiff=0.00000).
Runtime in cpu-seconds: 0.01
Number of SV: 2 (including 0 at upper bound)
L1 loss: loss=0.00000
Norm of weight vector: |w|=0.70711
Norm of longest example vector: |x|=4.24264
Estimated VCdim of classifier: VCdim<=10.00000
Computing XiAlpha-estimates...done
Runtime for XiAlpha-estimates in cpu-seconds: 0.00
XiAlpha-estimate of the error: error<=100.00% (rho=1.00,depth=0)
XiAlpha-estimate of the recall: recall=>0.00% (rho=1.00,depth=0)
XiAlpha-estimate of the precision: precision=>0.00% (rho=1.00,depth=0)
Number of kernel evaluations: 29
Writing model file...done
```

svm_classify test.dat file_model file_prediction

```
Reading model...OK. (2 support vectors read)
Classifying test examples..done
Runtime (without IO) in cpu-seconds: 0.00
Accuracy on test set: 100.00% (2 correct, 0 incorrect, 2 total)
Precision/recall on test set: 100.00%/100.00%
```

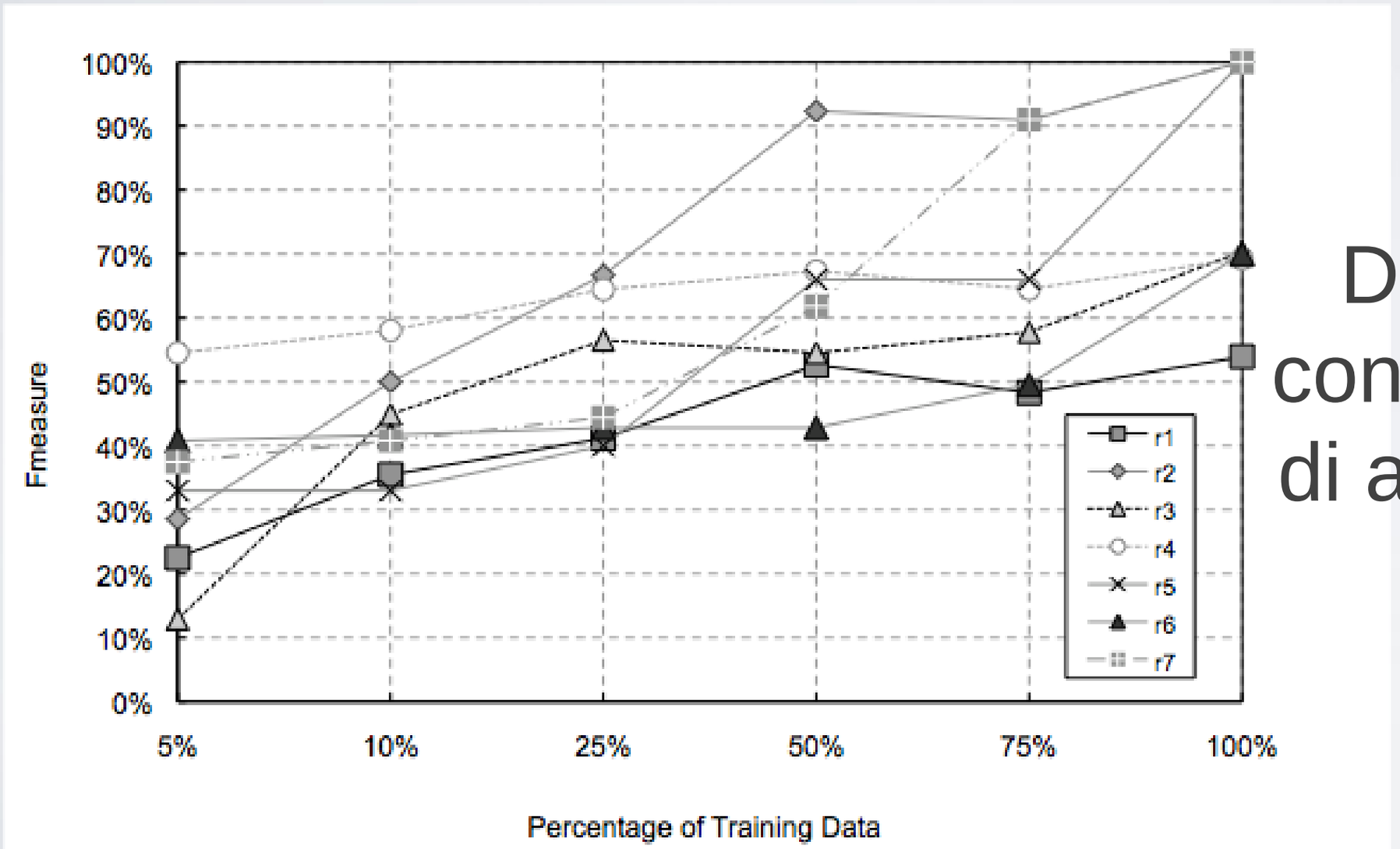
Valutazione prestazioni

Metriche

- Precision : TP/ FP+TP
- Recall: TP / FN+TP
- F-measure : 2PR/ P+R
- Accuracy: TP+TN/ TP+TN+FP+FN

Output analisys:

LEarning curve



Obbiettivo:
Determinare il contributo del dati di apprendimento

Multiclassification

- In alcuni casi può essere necessario classificare un dato tra più classi possibili
- Es: Classificare una persona come in_forma, sovrappeso o obesa
- problema: SVM è un classificatore binario

Multiclassification
the ONE VS ALL schema

- Mettere in competizione più svm tra loro:
  - Apprendimento: Addestramento delle singole con gli stessi dati di esempio, in cui: gli esempi positivi per una classe saranno negativi per le altre
  - Classificazione: Il dato viene classificato usando tutti i singoli modelli, e vengono confrontati gli output (file prediction), si sceglie il classificatore che ha etichettato il dato con il valore di distanza dal margine piu alto (positivo)

Support Vector Machines Kernel Function

- Qualora i dati non sia linearmente separabili si rende necessario mappare i dati di input in uno spazio opportuno dove possano essere separati con un iperpiano (*kernel trick*)
- esempi di kernel:
  - Lineare: $K(xi,xj)=<xi\cdot xj>$
  - Polinomiale: $K(xi,xj)=(<xi\cdot xj>+1)^d$ con $d \in IN$, $d \neq 0$
  - Gaussiano: $K(xi, xj) = \exp(-||xi - xj||^2/2\sigma)$

SVM Light

Kernel based learning

Kernel options:

-t int - type of kernel function:

0: linear (default)

1: polynomial (s a*b+c)^d

2: radial basis function exp(-gamma ||a-b||^2)

3: sigmoid tanh(s a*b + c)

4: user defined kernel from kernel.h

-d int - parameter d in polynomial kernel

-g float - parameter gamma in rbf kernel

-s float - parameter s in sigmoid/poly kernel

-r float - parameter c in sigmoid/poly kernel

-u string - parameter of user defined kernel

- Lineare: -t 0
- Polinomiale: -t 1-d 2
- Gaussiano: -t 2 -g 0.5

kernels for structured data

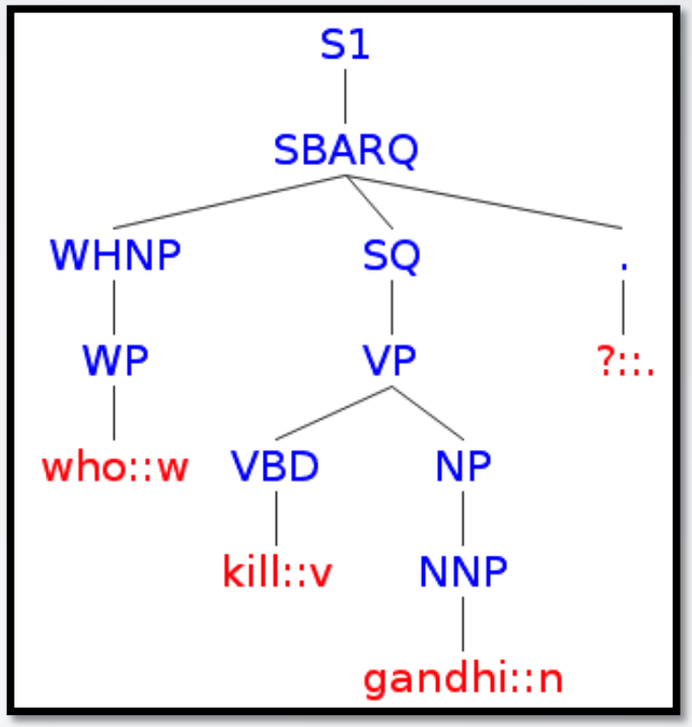
- Piattaforma SVM light TK : http://disi.unitn.it/~moschitt/Tree-Kernel.htm

- Oltre che con l’ uso di feature numeriche è possibile computare la funzione di classificazione anche su dati complessi:
  - sequenze,
  - alberi

Tree Kernels

Modeling syntax in Natural Language learning task is complex, e.g.

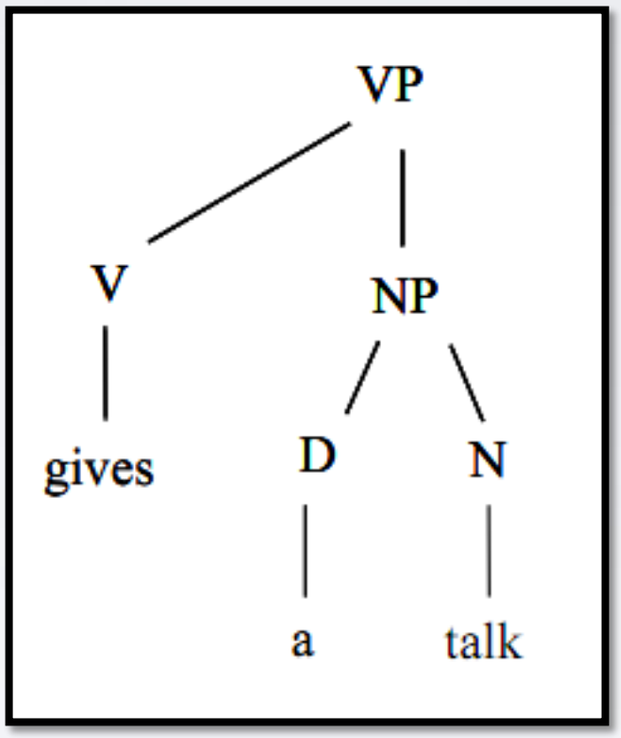
- Semantic role relations within predicate argument structures and
- Question Classification



Tree kernels are natural way to exploit syntactic information from sentence parse trees

- useful to engineer novel and complex features.

The Collins and Duffy's Tree Kernel

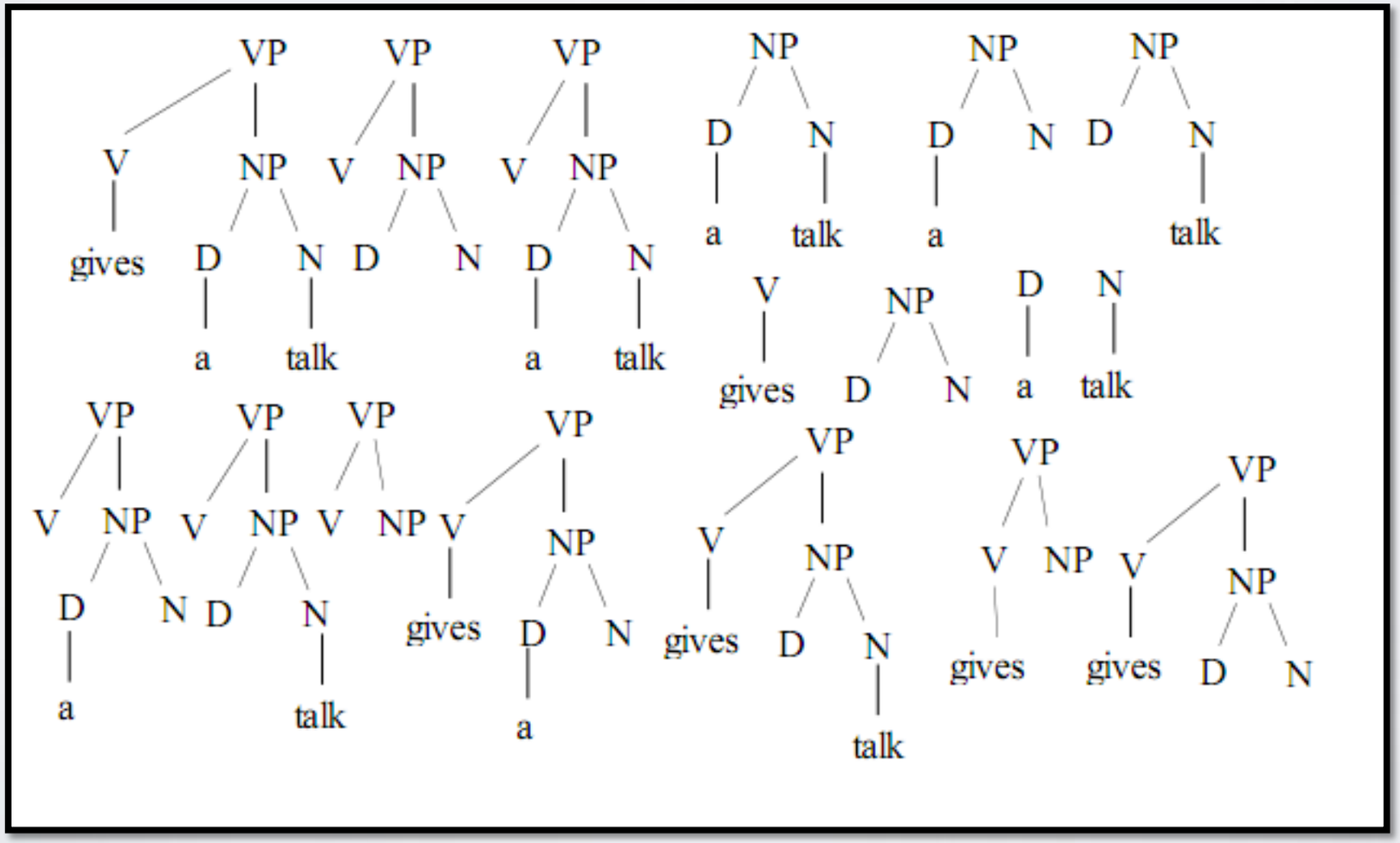


Given a costituency tree

The overall fragment set

We can explode the syntactic tree in *all* ***syntactically motivated*** fragments

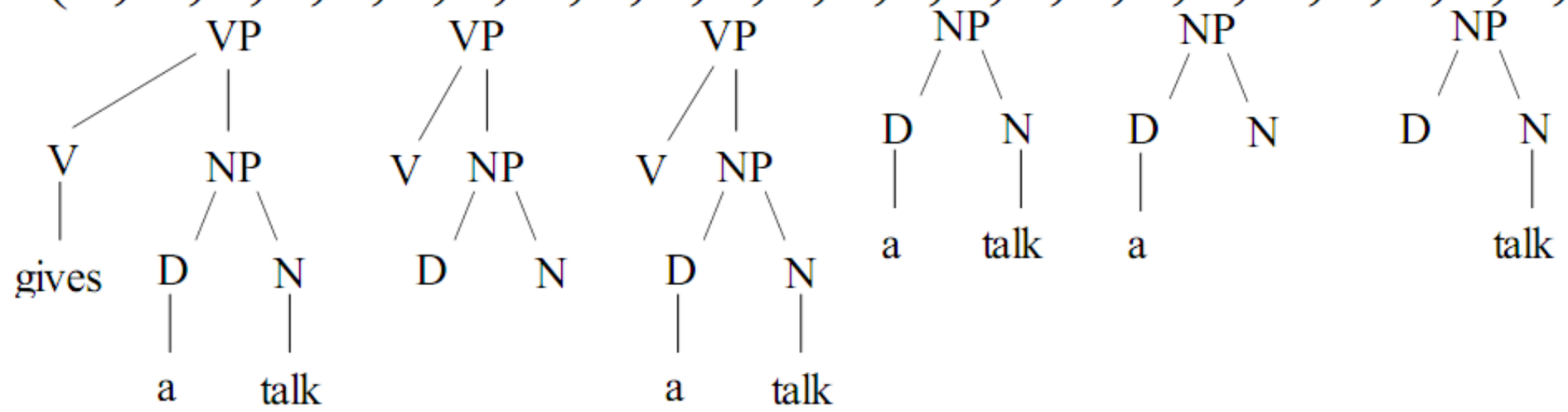
- For each node the production rules must be respected, i.e. we can remove “0 or all children at a time”
- It is also known as Syntactic Tree Kernel



Explicit feature space

Can we build a feature vector accounting on all this information?

$\vec{x}$ = (0, ..,1,..,0,..,1, ..,0,..,1,..,0, .,1,..,0,..,1, ..,0,..,1,..,0)



$\vec{x}_1 \cdot \vec{x}_2$ counts the number of common substructures

Implicit Representation

Can we estimate the tree kernel in an implicit space?

- We can implicitly count the number of common subtrees
- We prevent to define feature vectors that consider ALL POSSIBLE SUBTREES, i.e. thousand of features
- The final model will not contain feature vectors, but TREES

$$\vec{x}_1 \cdot \vec{x}_2 = \phi(T_1) \cdot \phi(T_2) = K(T_1, T_2) = \sum_{n_1 \in T_1} \sum_{n_2 \in T_2} \Delta(n_1, n_2)$$

[Collins and Duffy, ACL 2002] evaluate Δ in $O(n^2)$:

$\Delta(n_1, n_2) = 0$, **if the productions are different else**

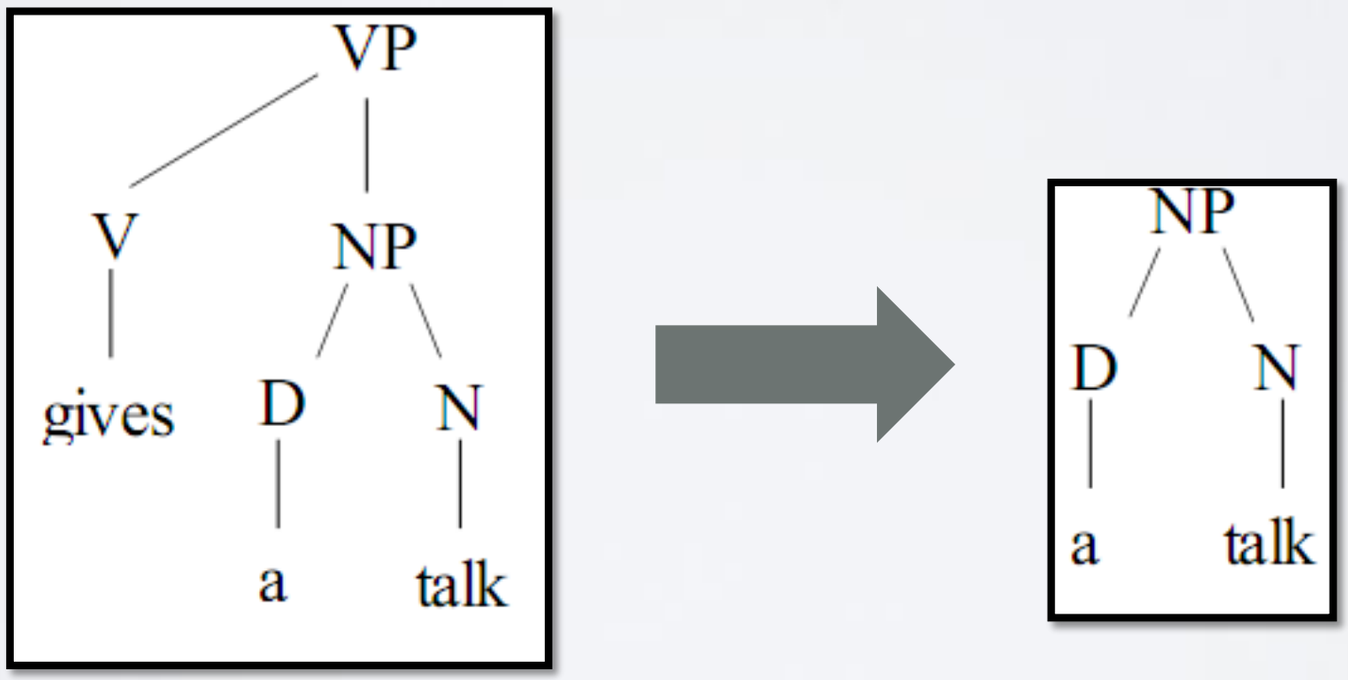
$\Delta(n_1, n_2) = 1$, **if pre-terminals else**

$$\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch(n_1, j), ch(n_2, j)))$$

Weighting

In the kernel estimation different subtrees are taken in account different times

- Es: in the following trees, one fragment will contribute twice to the overall kernel



Weighting

- A decay factor can be used, so the contribution of the embedded trees is reduced.
- The normalization of Tree Kernel estimation corresponds to the normalization of the explicit feature vector

Decay factor

$$\Delta(n_1, n_2) = \lambda, \quad \textbf{if pre-terminals else}$$

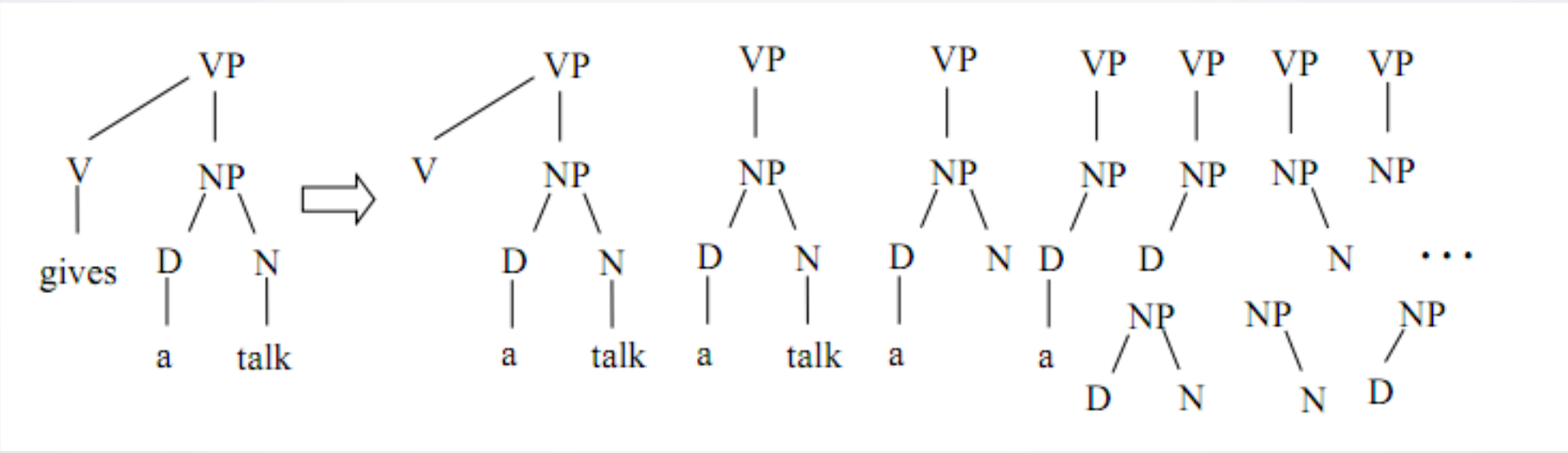
$$\Delta(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch(n_1, j), ch(n_2, j)))$$

Normalization

$$K'(T_1, T_2) = \frac{K(T_1, T_2)}{\sqrt{K(T_1, T_1) \times K(T_2, T_2)}}$$

Partial Tree [Moschitti,2006]

- A Syntactic Tree satisfies the constraint “remove 0 or all children at a time”.
- Partial Tree Kernel (PTK) relaxes such constraint we get more general substructures
- It allows gas in the production rules in the same fashion of the sequence kernel



Kernel Combination

A Tree Kernel can be still combined with a vector based kernel, in order to augment the information provided to the learning algorithm

Es: $K_{ij}=TK_1(t_i,t_j) + TK_2(t_i,t_j) + POLYN\text{-}K(x_i,x_j)$

```
-1 |BT| (TREE (ARG0 (A1 NP))(ARG1 (AM-NEG RB))(ARG2 (rel
fall))(ARG3 (AM-TMP NNP))(ARG4 (AM-TMP SBAR))(ARG5 null)(ARG6
null))    |BT| (TREE (ARG0 (A1 NP))(ARG1 (AM-NEG RB))(ARG2 (rel
fall))(ARG3 (A4 RP))(ARG4 (AM-TMP NNP))(ARG5 (AM-TMP SBAR))(ARG6
null)) |ET| 1:1 21:2.742439465642236E-4 23:1 30:1 36:1 39:1 41:1
46:1 49:1 66:1 152:1 274:1 333:1|EV|
```

Esercitazione

1) Trasformazione dataset (iris.arff) ARFF in file svm_ligth

2) Effettuare la multiclassificazione secondo lo schema One VS All