Perceptron

$$f(x) = \begin{cases} 1 & \text{if } w^T x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Data $\Rightarrow \{(x_i, y_i)\}$ x is $D \times 1$ w is $D \times 1$

Training

$\boxed{\text{INIT}}$ $w = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}}_{D}$ α = learning rate

```
while (!converged) {
    take some (x_i, y_i)
    if f(x_i) != y_i
        w <- w + α(f(x_i) - y_i) x_i
}
```

$$f(x_i) - y_i = \begin{cases} 1 & \text{if } y_i = 0 \\ -1 & \text{if } y_i = 1 \end{cases}$$

$w = \sum_{i=0}^{N} \beta_i x_i$ init $\beta = \begin{array}{cc} & 1 \\ N & \begin{bmatrix} 0 \\ 0 \\ \vdots \end{bmatrix} \end{array}$

update one β_i with each w update training requires 2 things

1. calc $f(x_i)$ for comparison w/ y_i
2. calc $\alpha(f(x_i) - y_i)$ to update β_i

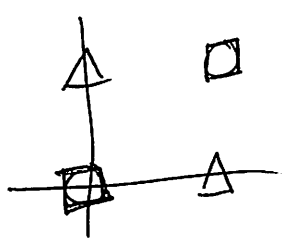
both only require $f(x_i)$, which requires only $w^T x_i$

$$w^T x = \left(\sum_{i=0}^{N} \beta_i x_i^T\right) x = \sum_{i=0}^{N} \beta_i \underbrace{(x_i^T x)}_{\text{KERNEL}}$$

Picking a Kernel

Kernels add dimensions, but must be way to seperate data linearly

consider XOR, not linearly seperable



□ (0,0)
△ (0,1)
△ (1,0)
□ (1,1)

adding a x+y ⇒
(0,0,0)
(0,[illegible],1)
([illegible],0,1)
(1,1,2)

not separable

adding xy ⇒
(0,0,0)
(0,[illegible],[illegible])
(1,0,0)
(1,1,1)

not seperable

adding x+y+xy

⇓

(0,0,0)
(0,1,1)
(1,0,1)
(1,1,3)

seperable

Valid Kernels

if K_1, K_2 are valid kernels
these are also valid

$aK_1, \quad a \in R^+$

$K_1 + K_2$

$K_1 K_2$

e^{K_1}

String Kernels

S_1 GATCCATCAGGTAC

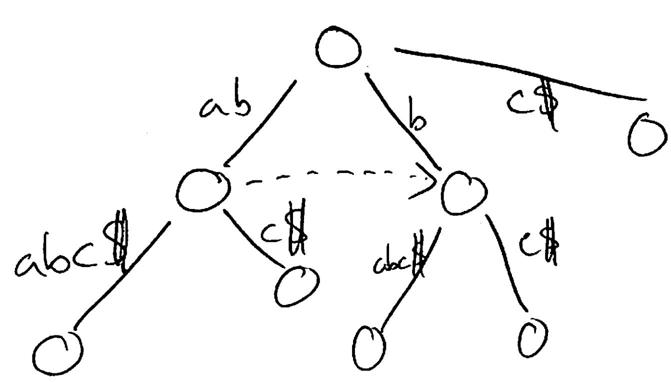
S_2 ACTACCATCGTACCA

$K(S_1, S_2) = \#$ of matching substrings

more general $\Rightarrow K(S_1, S_2) = \sum_{e \in SS} f(e)$ where SS is set of matching substrings

use suffix tree $\Rightarrow O(n)$

suffix tree for ababc



suffix links
- - - -> $O(n)$ time
give construction

what can we do with $f(e)$?

1. what if we want to consider everything at codon or longer level in DNA.

$$f = \begin{cases} 1 & \text{if } \operatorname{len}(e) \geq 3 \\ 0 & \text{otherwise} \end{cases}$$

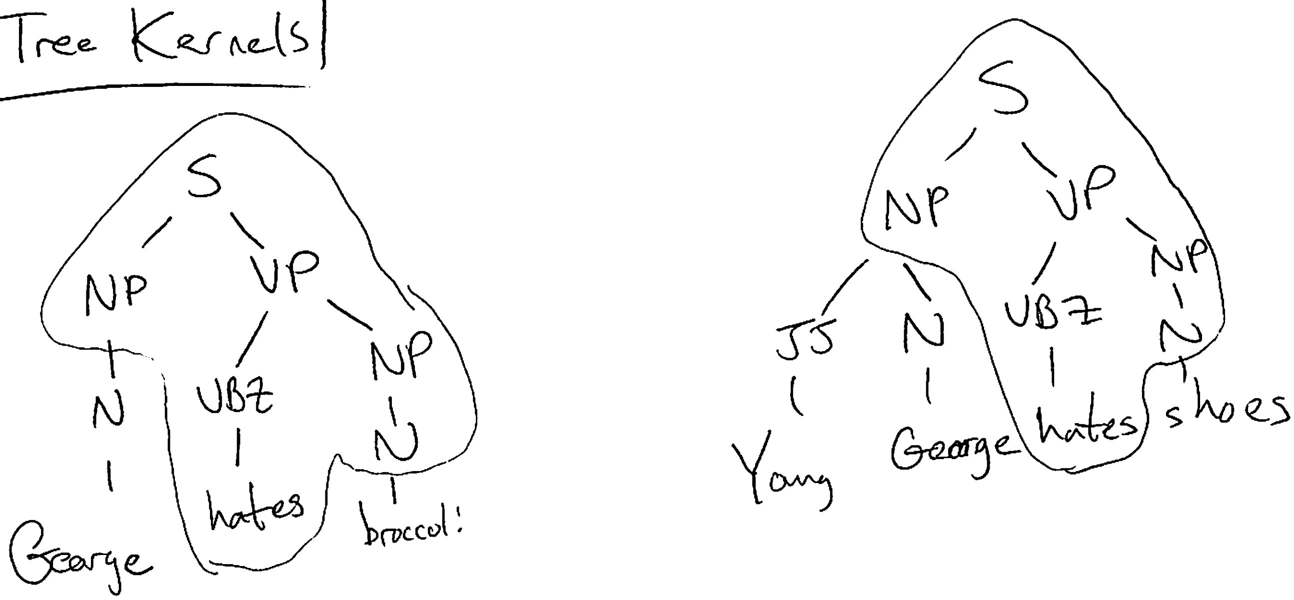
2. if we are considering native language, stopwords (and, the, or, etc...)

3. with e we actually have an alignment, so we can score the edit distance

eg $f(\text{theatre}, \text{theater}) > f(\text{theater}, \text{heater})$

do this by mapping word final re/er to a single new character, count their substitution with some weight $0 \leq w \leq 1$

Tree Kernels



	S	NP	JJ	N	VP	VBZ	NP	N
S	X							
NP		X					X	
N				X				X
VP					X			
VBZ						X		
NP		X					X	
N				X				X

1. find cells whose labels are equal (X's)
2. find cells in that set whose children all match (Ⓧ's)
3. for each Ⓧ, include it, and all children, recurse (3) on circled children

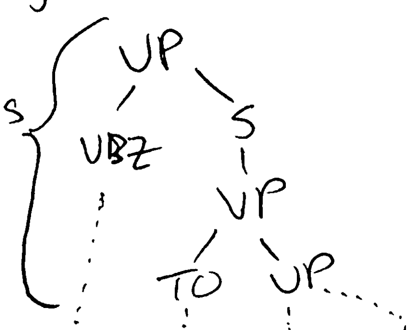
$$K = \sum_{e \in S} f(e)$$ S is the extracted set of trees

f=1 usually (counts)

Why tree kernels?

Given text in english, can we predict the native language?

evidence such as



This seems to be found in French

indicates french, corresponds to verb + infinitive combination often found in Native French