Perceptron

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

Data \( \mathcal{D} = \{(x_i, y_i)\} \), \( x \) is \( D \times 1 \) \( w \) is \( D \times 1 \)

\[ \text{Training} \]

\[ w = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \]

\( \alpha = \text{learning rate} \)

\[ \text{while} \text{(! converged)} \quad \exists \]

\[ \text{take some } (x_i, y_i) \]

\[ \text{if } f(x_i) \neq y_i \]

\[ w \leftarrow w + \alpha (f(x_i) - y_i) x_i \]

\[ \forall \]

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

\[ w = \sum_{i=0}^{N} \beta_i x_i \quad \text{init } \beta = \begin{bmatrix} 1 \\ 0 \quad \vdots \\ 0 \end{bmatrix} \]

update one \( \beta_i \) with each \( w \) update
Training requires 2 steps:
1. calc $f(x_i)$ for comparison w/ $y_i$
2. calc $\alpha (f(x_i) - y_i)$ to update $\beta_i$
both only require $f(x_i)$, which requires only $w^TX_i$

$$w^TX_0 = \left( \sum_{i=0}^{N} \beta_i x_i^T \right) X = \sum_{i=0}^{N} \beta_i (x_i^T X)$$

↑
KERNEL

Picking a Kernel

Kernels add dimensions, but must be used to separate data linearly
Consider XOR, not linearly separable

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$x+y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>(1, 0)</td>
<td>(0, 1)</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
</tr>
</tbody>
</table>

Adding $x+y$: not separable

Adding $xy$: not separable

Adding $x+y+xy$: separable
Valid Kernels:

- If \( K_1, K_2 \) are valid kernels, these are also valid:
  - \( aK_1, a \in \mathbb{R}^+ \)
  - \( K_1 + K_2 \)
  - \( K_1K_2 \)
  - \( eK_1 \)

String Kernels:

\[ \begin{align*}
S_1 & \rightarrow \text{GATCCATCAGGGTAC} \\
S_2 & \rightarrow \text{ACTACCATC GTACCA} \\
\end{align*} \]

\[ K(S_1, S_2) = \# \text{ of matching substrings} \]

More general: \( K(S_1, S_2) = \sum_{e \in \text{SS}} f(e) \) where \( \text{SS} \) is set of matching substrings

Use suffix tree \( \Rightarrow O(n) \)

Suffix tree for ababc

```
ab
/
O
O
```

Suffix links give O(n) time construction
what can we do with \( f(e) \)?

1. what if we want to consider everything at codon or longer level in DNA
   \[ f = \begin{cases} 1 & \text{if } \text{len}(e) \geq 3 \\ 0 & \text{otherwise} \end{cases} \]

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
   
   \[ \text{eg } f(\text{theatre, theater}) > f(\text{theater, heater}) \]
   
   do this by mapping word final to a single new character, count their substitution with some weight OSSW!

---

Tree Kernels

[Trees and diagrams illustrating different tree structures]
Why tree kernels?

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

Evidence such as

\[ \text{This seems to be found in French} \]

indicates French, corresponds to verb + infinitive combination often found in native French.

(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, recursively

\[ K = \sum_{e \in E} f(e) \]  
\( f=1 \) usually (counts)