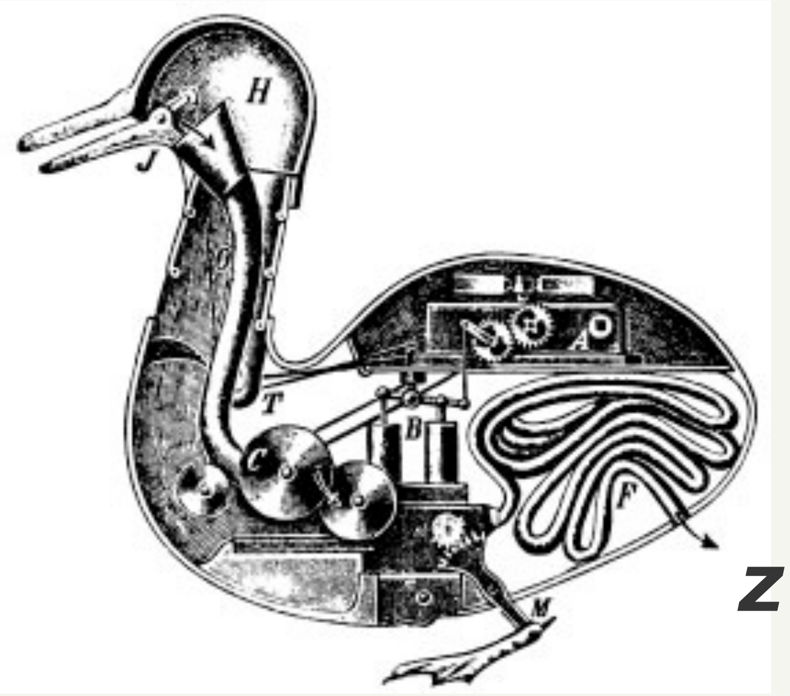
Some kernels for structured data

- Goal: construct a similarity score for object such as
  - sequences
    - with variable length
    - by their interpretations
  - labeled graphs (or trees)
    - different size
    - different structure
  - other objects
    - by their interpretations

- The similarity must be a PD kernel

Rational kernels

- Compare sequences $\boldsymbol{x}, \boldsymbol{y} \in \Sigma^*$
  - $\boldsymbol{x} = (0, 1, 1, 0, 1, 1, 0, 0)$
  - $\boldsymbol{y} = (1, 1, 0, 1)$
- <u>*Transducer*</u>
  - maps a seq. ***x*** sto seq. **z** with a weight
  - defines a "weighed relation" $T(\mathbf{x}, \mathbf{z}) \to \mathrm{R}$
  - is implemented by a *finite state automaton*



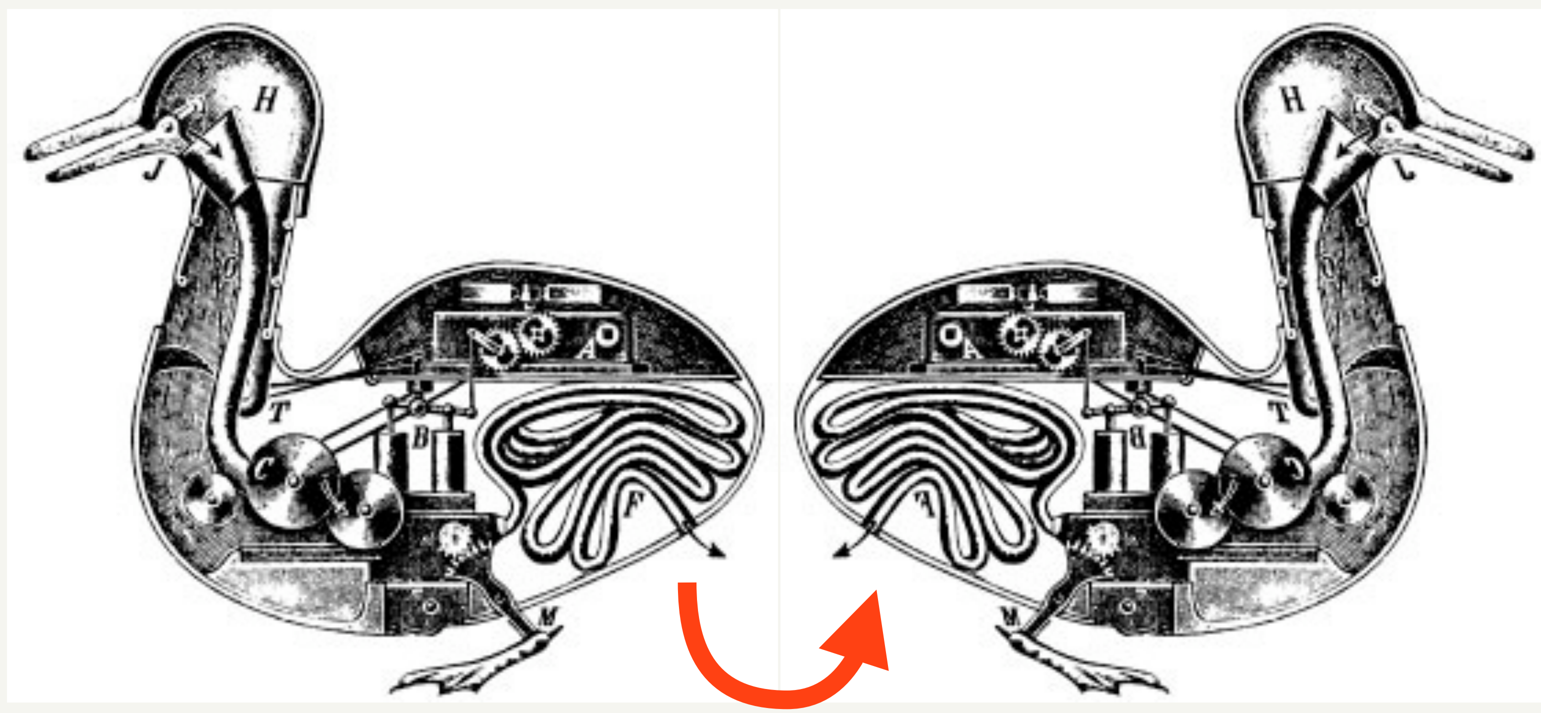
- Kernel
  x, ***y*** are similar if they are transduced often to the same ***z***
  - $K(\boldsymbol{x},\boldsymbol{y}) = \sum_{\boldsymbol{z}} T(\boldsymbol{x}, \boldsymbol{z})\, T(\boldsymbol{y}, \boldsymbol{z})$
- Advantage
  Given an automaton for T, can construct an automaton for K

Rational kernels: Implementation

- Automaton for $K(\boldsymbol{x}, \boldsymbol{y})$
  - invert T
  - compose T and T^{-1}

$$T \circ T^{-1}$$

x



y

z

Rational kernels: Examples

- Bag-of-subsequences
  - $\boldsymbol{x}$ binary sequence
  - $\boldsymbol{z}$ binary sequence of 4 characters
  - $T(\boldsymbol{x}, \boldsymbol{z})$ = # occurrences of $\boldsymbol{z}$ in $\boldsymbol{x}$
  - $K(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{z}} T(\boldsymbol{x}, \boldsymbol{z})\, T(\boldsymbol{y}, \boldsymbol{z})$ is large iff $\boldsymbol{x}, \boldsymbol{y}$ contain similar subsequences

$\boldsymbol{x} = (0, 1, 1, 0, 1, 1, 0)$

T	$\boldsymbol{z}$
2	0, 1, 1, 0
1	1, 1, 0, 1
1	1, 0, 1, 1

$\boldsymbol{y} = (1, 1, 0, 1)$

T	$\boldsymbol{z}$
1	1, 1, 0, 1

- Normalization
  $K(\boldsymbol{x}, \boldsymbol{y}) / (\, K(\boldsymbol{x}, \boldsymbol{x})\, K(\boldsymbol{y}, \boldsymbol{y})\,)^{1/2}$

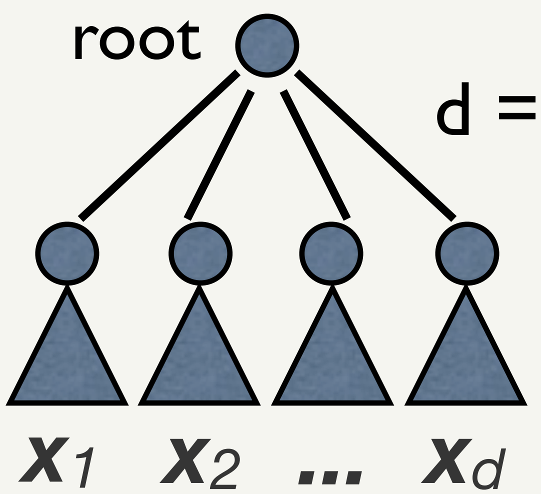
$K(\boldsymbol{x}, \boldsymbol{y}) = 1$

- Other examples
  - HMM-like models

Convolution kernels

- To compare objects ***x***, ***y***
  - decompose each object in *d* components
  - compare components and combine results
  - (repeat recursively until atomic components)
- Example: tree

x is a *d*-degree tree



Subpart relation

$R(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d, \mathbf{x})$

$$K(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{d} K_i(\mathbf{x}_i, \mathbf{y}_i)$$

Convolution kernels

- Example: string
  - ***x*** is a string
  - Subpart relation
    R($\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}$) iif
    $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ are (non-empty) strings such that $\boldsymbol{x}$ = concat($\boldsymbol{x}_1$, $\boldsymbol{x}_2$)

- Multiple decompositions are possible
  - $R^{-1}(\boldsymbol{x}) = \{ (\boldsymbol{x}_1, \boldsymbol{x}_2) : R(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}) \}$

- <u>**Convolution kernel**</u>

$$k(x, y) = \sum_{x' \in R^{-1}(x)} \sum_{y' \in R^{-1}(y)} \prod_{i=1}^{r} k_i(x'_i, y'_i)$$

Convolution kernels

We can represent the relation "$x_1, \ldots, x_d$ are the parts of x" by a relation R on the set $X_1 \times \cdots \times X_D \times X$, where $R(x_1, \ldots, x_D, x)$ is true iff $x_1, \ldots, x_D$ are the parts of x. For brevity, let $\vec{x} = x_1, \ldots, x_D$, and denote $R(x_1, \ldots, x_D, x)$ by $R(\vec{x}, x)$. Let $R^{-1}(x) = \{\vec{x} : R(\vec{x}, x)\}$. We say R is *finite* if $R^{-1}(x)$ is finite for all $x \in X$. Here are some examples:

1. If x is a D-tuple in $X = X_1 \times \cdots \times X_D$, and each component of $x \in X$ is a part of x, then $R(\vec{x}, x)$ iff $\vec{x} = x$.

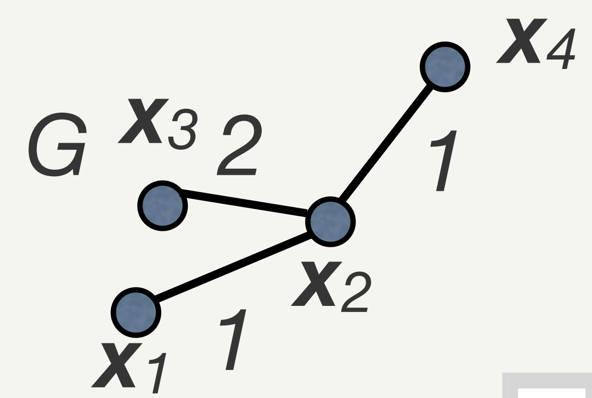
2. If $X_1 = X_2 = X$, where X is the set of all finite strings over a finite alphabet $\mathcal{A}$, then we can define $R(x_1, x_2, x)$ iff $x_1 \circ x_2 = x$, where $x_1 \circ x_2$ denotes the concatenation of strings x_1 and x_2.

3. Continuing the previous example, if the alphabet $\mathcal{A}$ has only one letter, then a finite string can be represented by the nonnegative integer n that is its length, so $X_1 = X_2 = X = \{0, 1, \ldots\}$ and $R(n_1, n_2, n)$ iff $n_1 + n_2 = n$.

4. If $X_1 = \ldots = X_D = X$, where X is the set of all D-degree ordered and rooted trees, then we can define $R(\vec{x}, x)$ iff $x_1, \ldots, x_D$ are the D subtrees of the root of the tree $x \in X$.

Kernels based on local info

- Given
  - { $\mathbf{x}_1, \dots, \mathbf{x}_n$ } collection of objects
  - "local" distances
    formally: G undirected weighed DAG



D	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$
$\mathbf{x}_1$	0	1	3
$\mathbf{x}_2$		0	2
$\mathbf{x}_3$			0

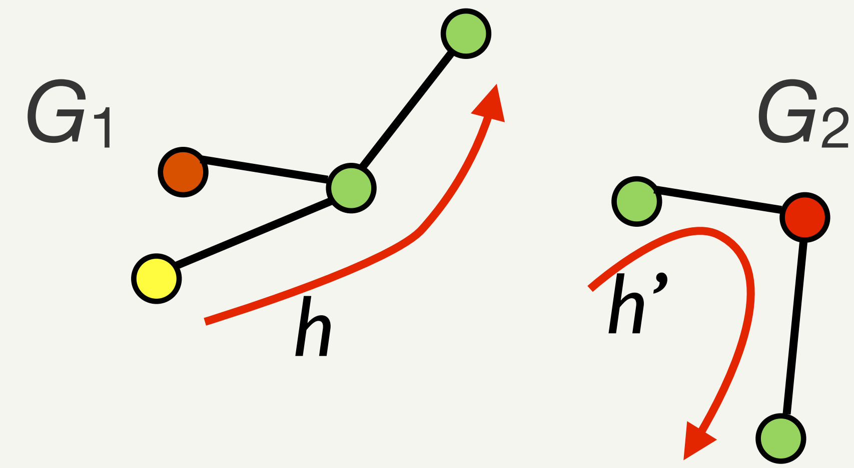
- Get geodesic distances D
  - all shortest-paths D
  - regularize by finding low-dimensional embedding (ISOMAP)

- Get a kernel
  - Use identity $D(\mathbf{x}_1,\mathbf{x}_2) = K(\mathbf{x}_1,\mathbf{x}_1) + K(\mathbf{x}_2,\mathbf{x}_2) - 2K(\mathbf{x}_1,\mathbf{x}_2)$
  - Make *positive definite* by incrementing the diagonal
    $K \leftarrow K + \lambda I$

See references in
Vedaldi Soatto "Quick Shift and Kernel Methods for Mode Seeking", ECCV, 2008

Graph kernels

- Compare *labeled graphs* ***x***, ***y*** ∈ Σ*
  - given a kernel on *paths* $k_{\text{path}}(h, h')$
  - extend to kenel on graphs
  - try to capture "topology"



- Compare all paths $W(G_1)$, $W(G_2)$

$$k_G(G_1, G_2) = \sum_{h \in W(G_1)} \sum_{h' \in W(G_2)} k(h, h').$$

k_{path}

  - walks (any path)
  - proper paths (no self intersection)
  - shortest paths
  - random walks

- Suggested reading :-)
  Vedaldi Soatto, "Relaxed Matching Kernels", 2008

Fisher kernels

- Compare objects ***x***, ***y*** by a *generative model*
  - given p(***x*** | θ)
  - map points ***x*** to maximum-likelihood parameters $\theta_{\boldsymbol{x}}$
  - compare $K(\theta_{\boldsymbol{x}}, \theta_{\boldsymbol{y}})$
- Local analysis
  - log-likelihood function $L(\mathbf{x}, \theta) = \log p(\mathbf{x}|\theta)$
  - assume $\mathbf{x} \sim p(\mathbf{x}|\theta)$
  - maximum likelihood is consistent $\forall \widehat{\theta} : \; E[L(x, \widehat{\theta})] \leq E[L(x, \theta)]$
- Fisher score

$$U(\mathbf{x}, \theta) = \nabla_\theta L(\mathbf{x}, \theta) \qquad E[U(\mathbf{x}, \theta)] = \frac{\partial}{\partial \theta} E[L(\mathbf{x}, \theta)] = 0$$

- Fisher information

$$I(\theta) = E[U(\mathbf{x}, \theta)^2] = \operatorname{var} U(\mathbf{x}, \theta)$$

Fisher kernels

- Fisher information matrix as approx. second derivative

$$
\begin{aligned}
E\left[\frac{\partial^2}{\partial\theta^2}L(\mathbf{x},\theta)\right] &= E\left[\frac{1}{p(\mathbf{x}|\theta)}\frac{\partial^2}{\partial\theta^2}p(\mathbf{x}|\theta)\right] - E\left[\left(\frac{1}{p(\mathbf{x}|\theta)}\frac{\partial}{\partial\theta}p(\mathbf{x}|\theta)\right)^2\right] \\
&\approx -E\left[\left(\frac{\partial}{\partial\theta}\log p(\mathbf{x}|\theta)\right)^2\right] \\
&= -E[U(\mathbf{x},\theta)^2] = -I(\theta)
\end{aligned}
$$

- Approx. ML estimate

$$L(\mathbf{x},\theta+\delta\theta) \approx L(\mathbf{x},\theta) + U(\mathbf{x},\theta)\delta\theta - \frac{1}{2}I(\theta)(\delta\theta)^2$$

$$\delta\theta_{\mathbf{x}} \approx I(\theta)^{-1}U(\mathbf{x},\theta)$$

- Fisher kernel

$$K(\mathbf{x},\mathbf{y}) = \delta\theta_{\mathbf{x}}\, I(\theta)\, \delta\theta_{\mathbf{y}} = U(\mathbf{x},\theta)I(\theta)^{-1}U(\mathbf{y},\theta)$$

Invariance

- Why weighting by I ?

$$K(\mathbf{x},\mathbf{y}) = \delta\theta_{\mathbf{x}}\, I(\theta)\, \delta\theta_{\mathbf{y}} = U(\mathbf{x},\theta) I(\theta)^{-1} U(\mathbf{y},\theta)$$

- Reparametrization $\theta = \phi(\lambda)$

$$L'(\mathbf{x},\lambda) = L(\mathbf{x},\phi(\lambda)) \qquad U'(\mathbf{x},\lambda) = U(\mathbf{x},\phi(\lambda))\dot{\phi}(\lambda)$$

$$I'(\lambda) = \dot{\phi}(\lambda)\, I(\phi(\lambda))\, \dot{\phi}(\lambda)$$

- Fisher kernel is invariant to reparametrization

$$K(\mathbf{x},\mathbf{y}) = U'(I')^{-1}U' = U\phi\phi^{-1}I\phi^{-1}\phi U = UI^{-1}U$$

Tutorial

- MediaLandscape Player