Kernel Methods for Structured In- & Outputs

Thomas Gärtner

Supervised Classification Example

 $\mapsto +1$

 $\mapsto -1$

 $\mapsto -1$

 $\mapsto +1$

Supervised Classification Example

 $\mapsto +1$

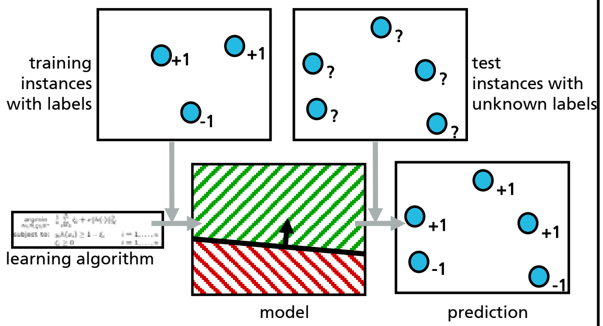
 $\mapsto -1$

 $\mapsto -1$

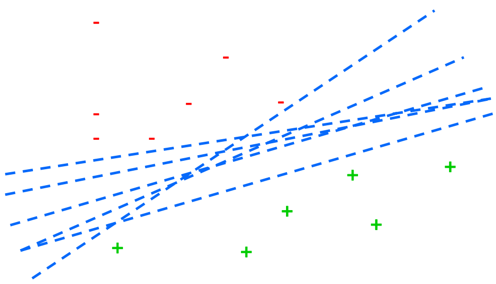
 $\mapsto +1$

 $\mapsto ?$

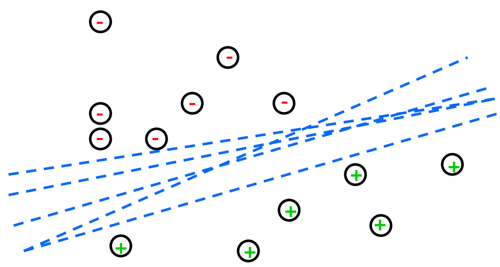
Supervised Classification



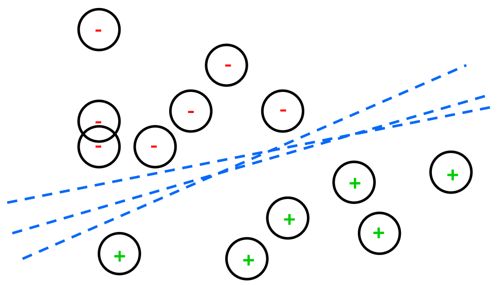
Classification by Linear Functions



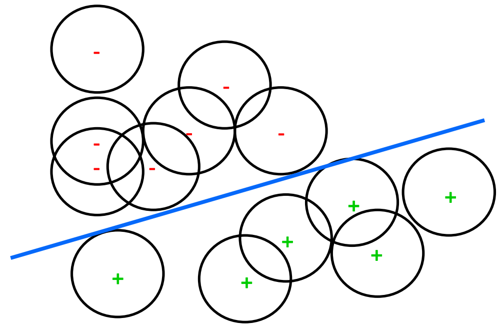
Motivation: Support Vector Machines



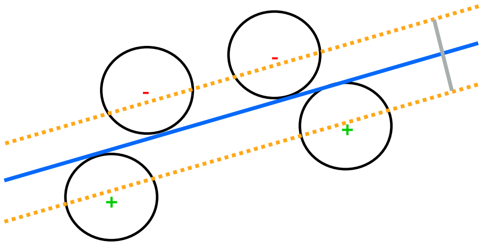
Motivation: Support Vector Machines



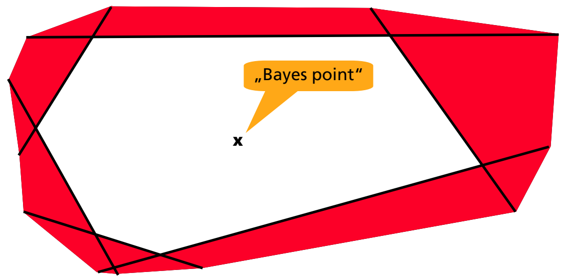
Motivation: Support Vector Machines



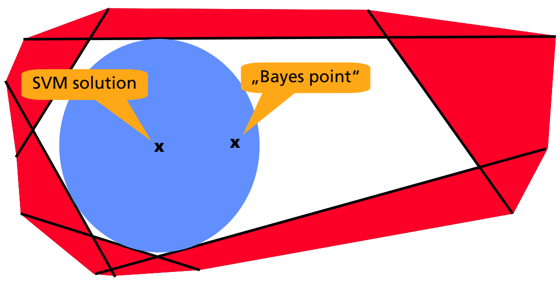
Motivation: Maximal Margin Hyperplane



Version Space Geometry of SVMs



Version Space Geometry of SVMs



Regularised Risk Minimisation Principle

$$\hat{h} = \underset{f \in \mathcal{F}}{\text{argmin}} \; \frac{1}{n} \sum_{i=1}^{n} V(y_i, f(x_i)) \quad \text{s.t.} \quad \Omega[f] \leq \lambda$$

error on training data (drawn iid)

capacity of hypothesis space

$$\hat{h} = \underset{f \in \mathcal{F}}{\text{argmin}} \; \frac{1}{n} \sum_{i=1}^{n} V(y_i, f(x_i)) \; + \; \nu \Omega[f]$$

Regularised Risk Minimisation Principle

$$\hat{h} = \operatorname*{argmin}_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} V(y_i, f(x_i)) \quad \text{s.t.} \quad \Omega[f] \le \lambda$$

error on training data (drawn iid)

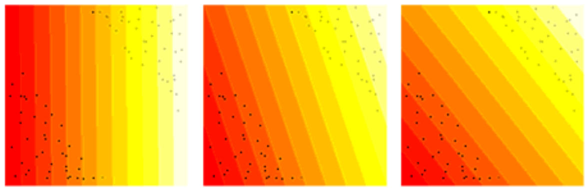
capacity of hypothesis space

$$\hat{h} = \operatorname*{argmin}_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} V(y_i, f(x_i)) + \nu\Omega[f]$$

kernel methods have commonly F: RKHS with pd kernel k; $\Omega[.] = \|.\|^2$; V convex

-> resulting optimisation problem is convex
-> minimiser lies in the span of $\{ k(x_i, .) \}_i$

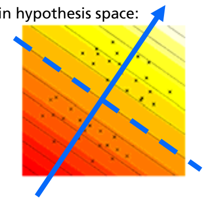
Stability with respect to Outliers



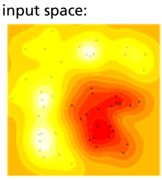
Kernel Methods

$$f(\cdot) = \sum c_i k(x_i, \cdot)$$

in hypothesis space:



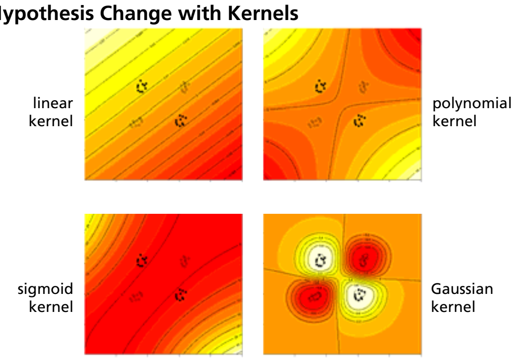
in input space:



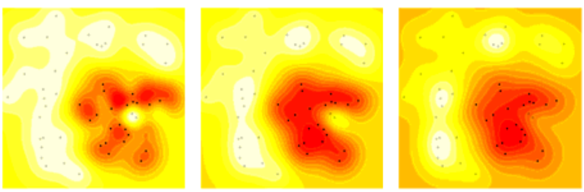
before, we were looking for a hyperplane in input space,
now, we are looking for a hyperplane in a function space (a RKHS with kernel k)
where each x is represented by the function k(x,.)

requiring an RKHS means basically everything works as before

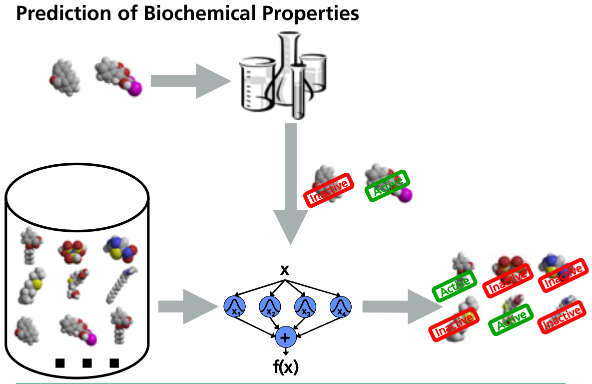
Hypothesis Change with Kernels



Stability with respect to Outliers



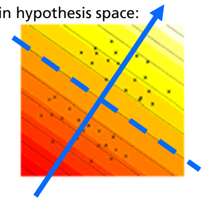
Prediction of Biochemical Properties



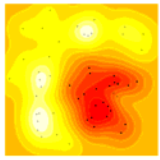
Approach: Kernel Methods for Graphs

$$f(\cdot) = \sum c_i k(x_i, \cdot)$$

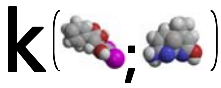
in hypothesis space:



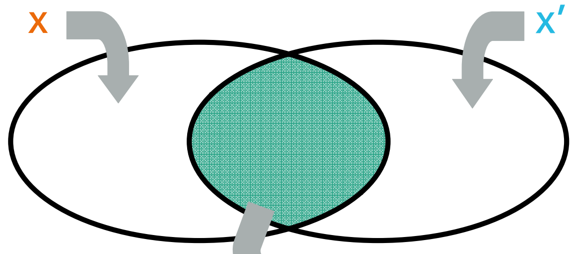
in input space:



we need valid kernel functions for molecules:



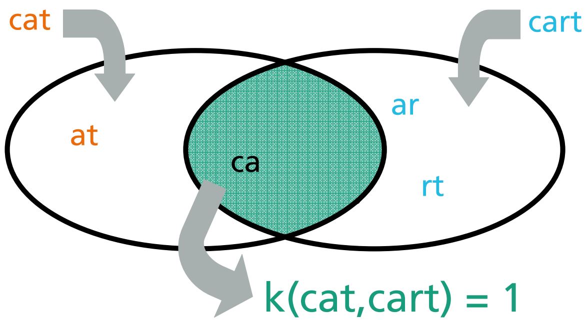
The "Intersection Kernel"-Principle (1)



can be feasible even if
x is only given intensionally
and even if |x| is exponential
in its representation!!!

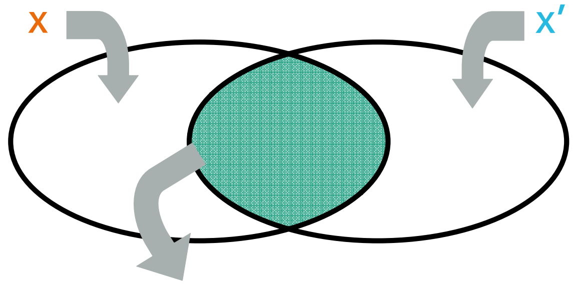
$$k(x,x')=\mu(x \cap x')$$

n-gram Kernels (n = 2)



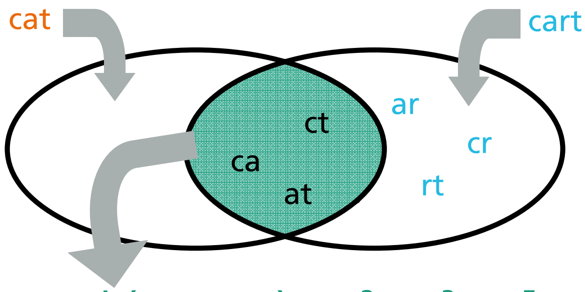
$$k(cat,cart) = 1$$

The "Intersection Kernel"-Principle (2)



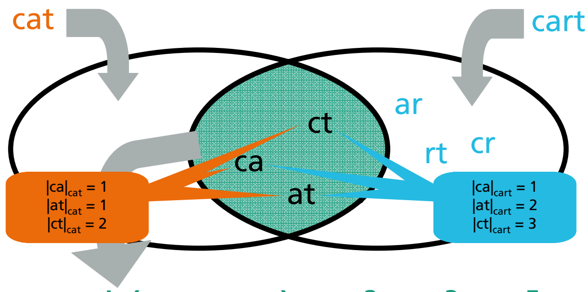
$$k(x,x')=\sum_{p \in x \cup x'} f(|p|_x) \cdot f(|p|_{x'})$$

Subsequence Kernels (n = 2)



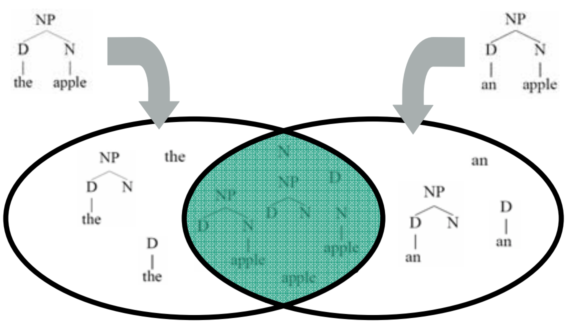
$$k(cat,cart) = \lambda^2 + \lambda^3 + \lambda^5$$

Subsequence Kernels (n = 2)

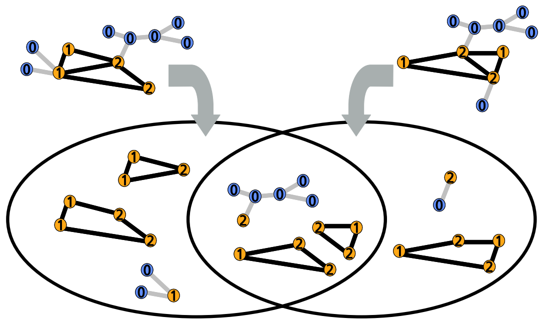


$$k(cat,cart) = \lambda^2 + \lambda^3 + \lambda^5$$

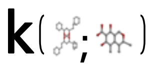
(Parse-) Tree Kernel

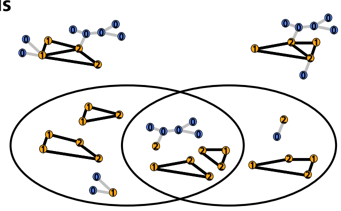


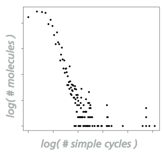
Graph Kernels



Cyclic Pattern Kernels







computing CPK is NP-hard
but efficient learning is still possible as
most compounds contain only a few cycles

`complete' as well as all-subgraphs graph kernels are hard
polynomial algorithm for all-walks kernels [colt'03]
(enumeration of) cyclic patterns [kdd'04]
walk kernel for RRL [mlj'06]
[ilp'03] "best algorithmic paper"

Graph Kernels --- Negative Results

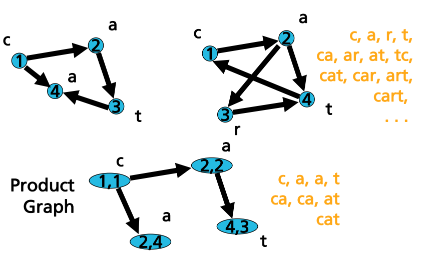
Computing the "all patterns" graph kernel is hard for instance for patterns that are "all paths", "all cycles", or "all connected subgraphs" with subgraph-isomorphism as the embedding operator.

Computing any graph kernel for which

$$k(G,\cdot) = k(G',\cdot) \Leftrightarrow G \simeq G'$$

is at least as hard as deciding graph isomorphism

All-Walks Kernels



$$k_\times(G_1,G_2) = 1^\top \left[\lim_{n\to\infty}\sum_{i=0}^{n}\lambda_i E_\times^i\right] 1 \underset{\text{eg}}{=} 1^\top(\mathbf{I}-\gamma E_\times)^{-1}1$$

Limits of matrix power series

eigenvalue decomposition

$$E = TDT^{-1} \Rightarrow \quad \lambda_i(E^n) = (\lambda_i(E))^n$$

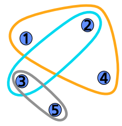
exponential series

$$\sum_{n=0}^{\infty} \frac{\beta^n}{n!} E^n \quad \Rightarrow \quad \lambda_i\left(\sum_n^{\infty} \frac{\beta^n}{n!} E^n\right) = e^{\beta\lambda_i(E)}$$

geometric series

$$\sum_{n=0}^{\infty} \gamma^n E^n \quad \Rightarrow \quad \lambda_i\left(\sum_n^{\infty} \gamma^n E^n\right) = \frac{1}{1-\gamma\lambda_i(E)}$$

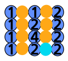
Kernels between Vertices in (Hyper-) Graphs



$v_1^\top v_2 = 0$

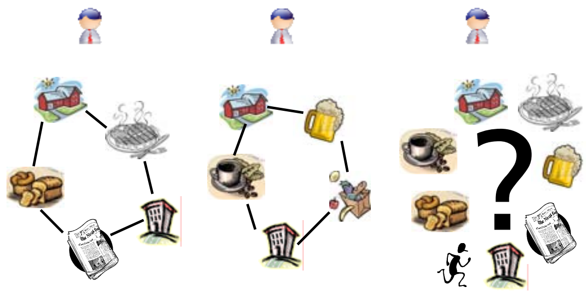
$v_1^\top BB^\top v_2 = 1$

$v_1^\top BB^\top BB^\top v_2 = 4$



$\left[\sum_{n=0}^{\infty} \lambda_n \left(BB^\top\right)^n\right]_{ij}$ weighted sum over walks from vertex *i* to *j* in the hypergraph.

Travelling John Q Public Problem



Structured Output Prediction

input & output spaces $\mathcal{X}, \mathcal{Y}(\Sigma)$

training data $D = \{(x_i, Y_i)\}_{i \in [\![n]\!]} \subseteq \mathcal{X} \times 2^{\mathcal{Y}(\Sigma)}$ (often $|Y_i| = 1$)

Ansatz

- joint scoring function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$
- **prediction / decoding** $g(x) = \operatorname{argmax}_{y \in \mathcal{Y}} f(x, y)$

Structured Output Prediction

input & output spaces $\mathcal{X}, \mathcal{Y}(\Sigma)$

training data $D = \{(x_i, Y_i)\}_{i \in [\![n]\!]} \subseteq \mathcal{X} \times 2^{\mathcal{Y}(\Sigma)}$ (often $|Y_i| = 1$)

Ansatz

- joint scoring function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$
- **prediction / decoding** $g(x) = \operatorname{argmax}_{y \in \mathcal{Y}} f(x, y)$

typically, the output space is exponential in |D|+|Σ|

Finding the Scoring Function (simplified)

$$\hat{f} = \operatorname*{argmin}_{f \in \mathcal{H}} \quad \Omega[f]$$
$$\text{subject to} \quad f(x_i, y) \; > \; f(x_i, z) \quad (\forall i, \forall z \in \mathcal{Y} \setminus Y_i, y \in Y_i)$$

has exponentially many constraints !!

Structured Perceptron

make as few as possible mispredictions on a sequence (x_1,y_1), (x_2,y_2), …

```
initialise f_0 <- 0
for x_i predict argmax_z f_{i-1}(x,z)
if misprediction
  then let f_i(.,.) <- f_{i-1}(.,.) + k((x_i,y_i),(.,.))
  else let f_i(.,.) <- f_{i-1}(.,.)
```

Structured Perceptron

make as few as possible mispredictions on a sequence (x_1,y_1), (x_2,y_2), …

```
initialise f_0 <- 0
for x_i predict argmax_z f_{i-1}(x,z)
if misprediction
  then let f_i(.,.) <- f_{i-1}(.,.) + k((x_i,y_i),(.,.))
  else let f_i(.,.) <- f_{i-1}(.,.)
```

Novikoff's Theorem applies and allows to bound the number of mispredictions as O(R² / δ²)

SVM-like Structured Output with Decoding-Oracle

$$\hat f = \operatorname*{argmin}_{f\in\mathcal{H}} \quad \nu\|f\|^2 + \|\xi_i\|_1$$
$$\text{subject to} \quad f(x_i,y) \; > \; \Delta(y,z) + f(x_i,z) - \xi_i \quad (\forall i, \forall z \in \mathcal{Y}\setminus\{y_i\})$$

can be solved efficiently by cutting-plane techniques

$$\hat f = \operatorname*{argmin}_{f\in\mathcal{H}} \quad \|w\|_1 + \|\xi_i\|_1$$
$$\text{subject to} \quad \langle w, \phi(x_i,y_i)\rangle \; > \; \Delta(y,z) + \langle w, \phi(x_i,z)\rangle - \xi_i$$

can be solved efficiently by the ellipsoid method $(\forall i, \forall z \in \mathcal{Y}\setminus\{y_i\})$

Finding Violated Constraints

decoding (polynomial time?)
- given f, x find $z \in \operatorname*{argmax}_{y\in\mathcal{Y}} f(x,y)$

separation (polynomial time?)
- given f, x, y find any $z : f(x,y) < f(x,z)$ or proof that none exists

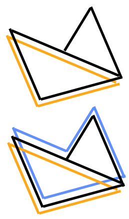
optimality (polynomial time? / in NP?)
- given f, x, y decide $\nexists z \in \mathcal{Y} : f(x,y) < f(x,z)$

"**decoding is in P**" is the strongest assumption while "**optimality is in NP**" is the weakest

Optimality vs Non-Optimality

optimality (in NP?) is there **no** longer cycle
- given f, x, y decide $\nexists z \in \mathcal{Y} : f(x,y) < f(x,z)$
- what is a short certificate of optimality?

sub-optimality (in NP!) is there **any** longer cycle
- given f, x, y decide $\exists z \in \mathcal{Y} : f(x,y) < f(x,z)$
- certificate of non-optimality is short

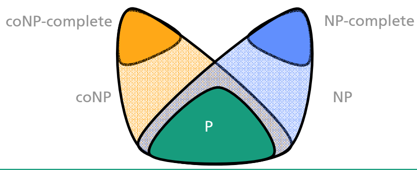


we are interested mostly in problems where **sub-optimality** is NP-complete

Optimality vs Non-Optimality

optimality (coNP-complete and hence not in NP)
- given f, x, y decide $\nexists z \in \mathcal{Y} : f(x,y) < f(x,z)$

sub-optimality (NP-complete)
- given f, x, y decide $\exists z \in \mathcal{Y} : f(x,y) < f(x,z)$



Structured Output Ranking

$h^* = \operatorname*{argmin}_{f:\mathcal{X}\times\mathcal{Y}\to\mathbb{R}}$ consider **regularised AUC** of **all ranked outputs**

- representer theorem $h^* \in \operatorname{span}$ **exponentially** many coefficients

Structured Output Ranking

$h^* = \operatorname{argmin}_{f:\mathcal{X}\times\mathcal{Y}\to\mathbb{R}}$ **consider regularised AUC** of all ranked **outputs**

- representer theorem $h^* \in \mathrm{span}$ **exponentially** many coefficients

assume a **tensor product of Hilbert spaces** $h \in \mathcal{H} = \mathcal{H}_{\mathcal{X}} \otimes \mathcal{H}_{\mathcal{Y}}$
assume a small output basis (d)

- **factorised representer theorem**

$h^* \in \mathrm{span}$ **polynomially** many coefficients

minimise a quadratic upper bound on the AUC

Structured Output Ranking

the following terms occur in the objective

$$\sum_{y\in\mathcal{Y}} f(x_i, y) = {f_\alpha^i}^\top \sum_{z\in\mathcal{Y}} \phi(z) = {f_\alpha^i}^\top \Phi$$

$$\sum_{y\in\mathcal{Y}} f^2(x_i, y) = {f_\alpha^i}^\top \sum_{z\in\mathcal{Y}} \phi(z)\phi^\top(z) f_\alpha^i = {f_\alpha^i}^\top \mathbf{C} f_\alpha^i$$

Structured Output Ranking

the following terms occur in the objective

$$\sum_{y\in\mathcal{Y}} f(x_i, y) = {f_\alpha^i}^\top \sum_{z\in\mathcal{Y}(\Sigma)} \phi(z) = {f_\alpha^i}^\top \Phi$$

$$\sum_{y\in\mathcal{Y}} f^2(x_i, y) = {f_\alpha^i}^\top \sum_{z\in\mathcal{Y}(\Sigma)} \phi(z)\phi^\top(z) f_\alpha^i = {f_\alpha^i}^\top \mathbf{C} f_\alpha^i$$

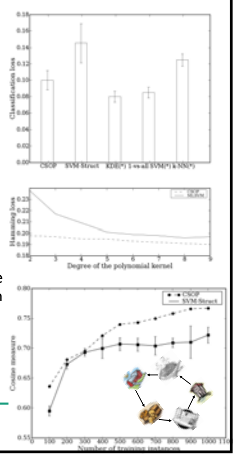
often these **exponential sums** can be computed in **polynomial time**

Summary

most approaches assume (at least) that **optimality** is in NP. However, it is often **co-NP complete**.

we proposed a **ranking** based approach that results in an **unconstrained convex optimisation problem.**

The new approach complements the existing one and is based on an orthogonal assumption. It can be adapted to new output spaces by coding one new matrix.



Constructive Machine Learning for De Novo Drug Design