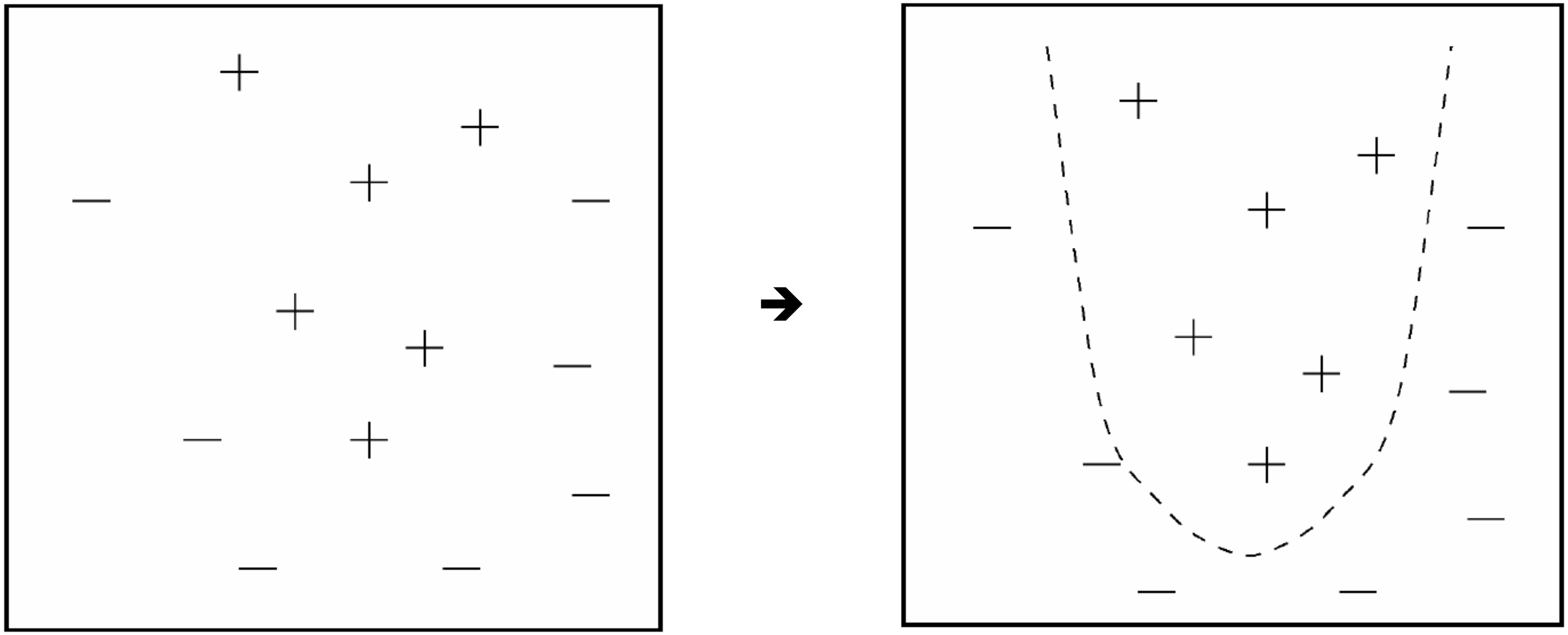
Support Vector Machines

Outline

- Transform a linear learner into a non-linear learner
- Kernels can make high-dimensional spaces tractable
- Kernels can make non-vectorial data tractable

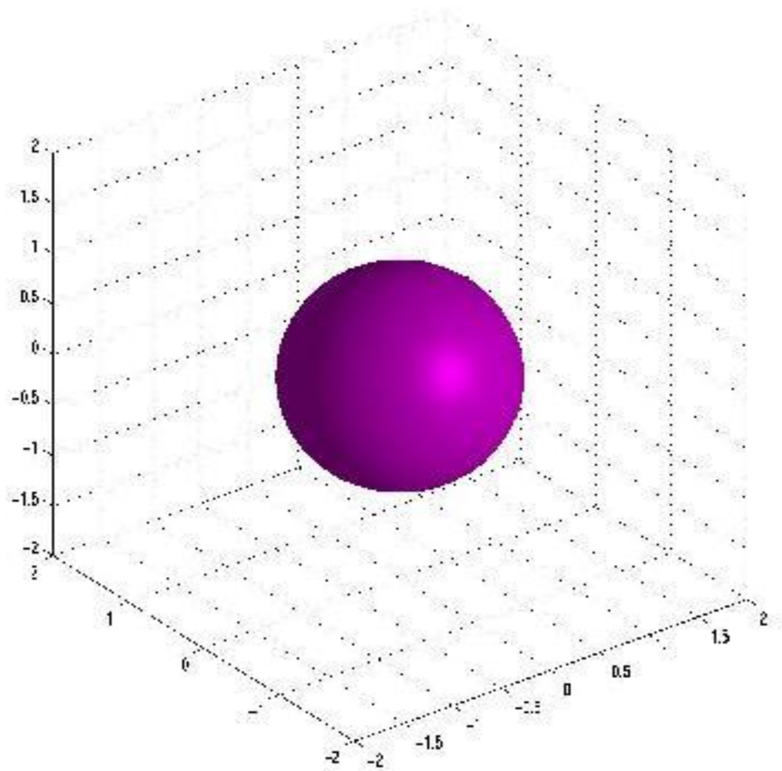
Non-Linear Problems

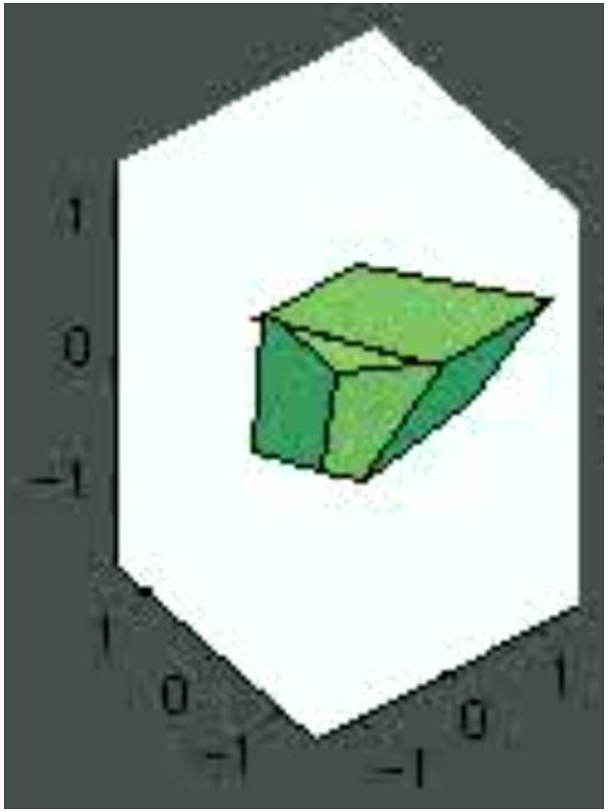


Problem:

- some tasks have non-linear structure
- no hyperplane is sufficiently accurate

How can SVMs learn non-linear classification rules?

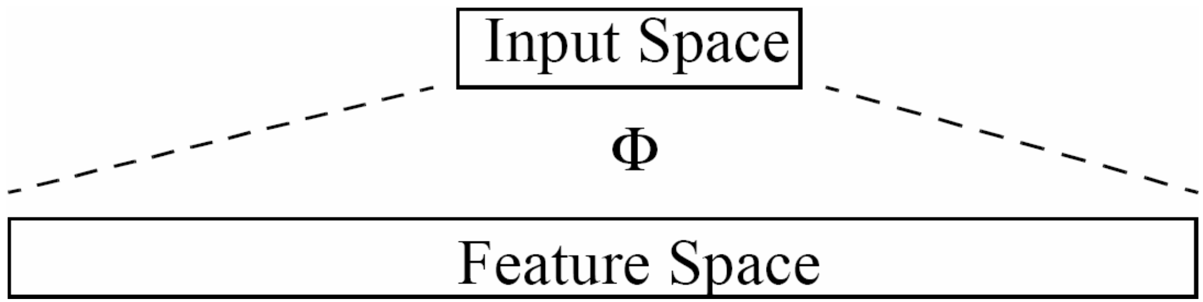




Ofer Melnik, http://www.demo.cs.brandeis.edu/pr/DIBA

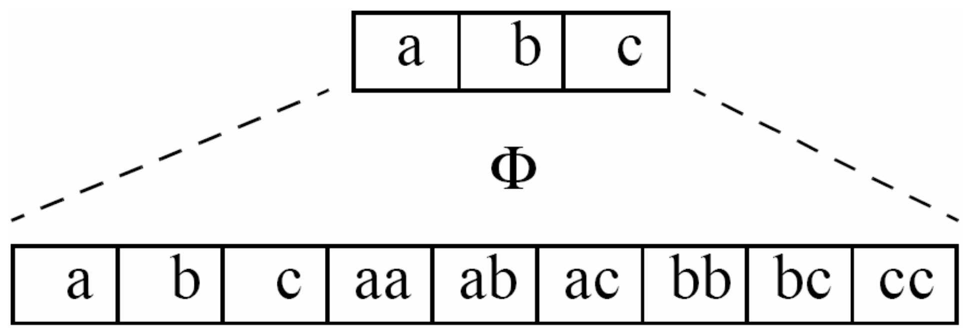
Extending the Hypothesis Space

Idea: add more features

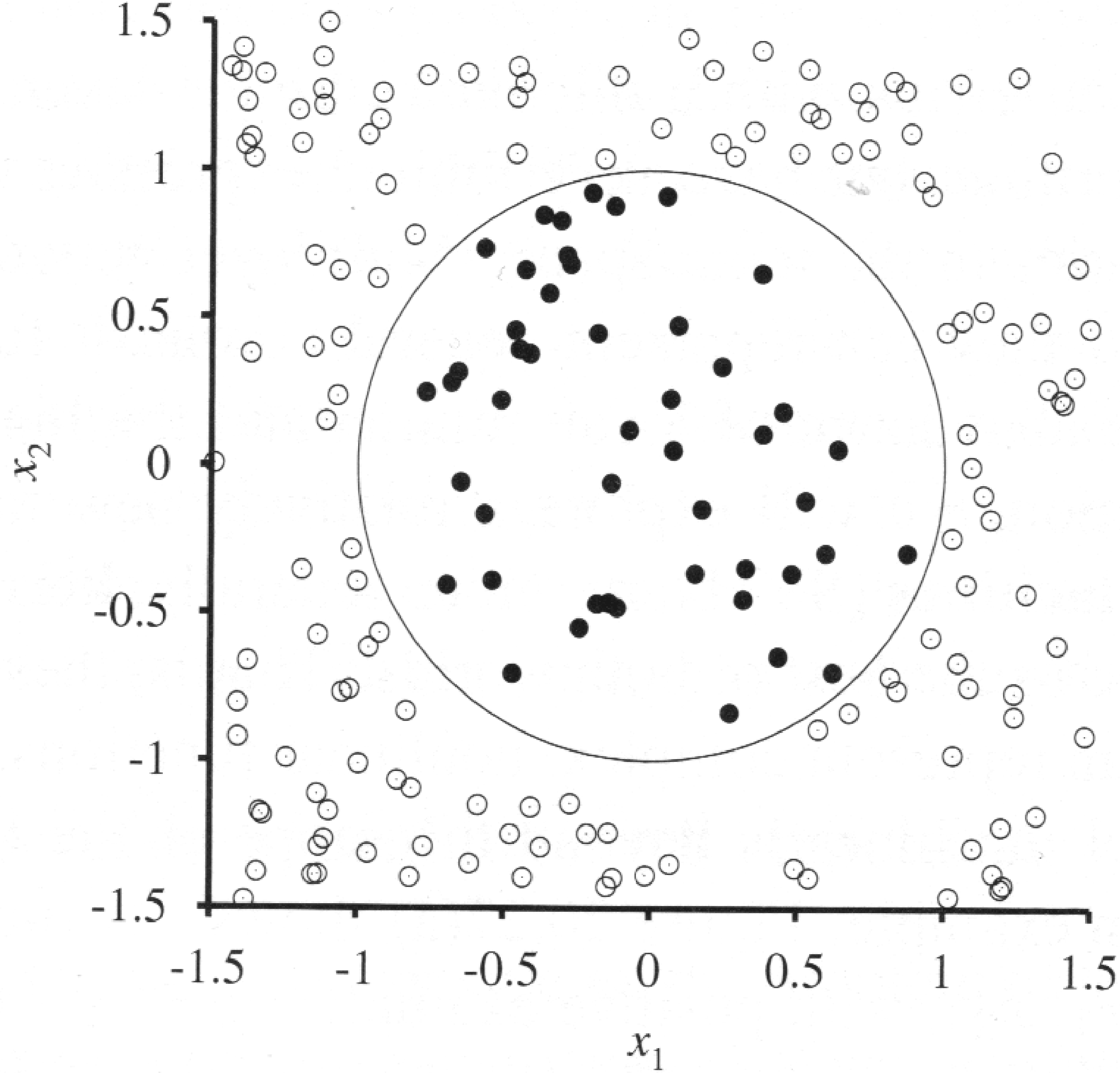


➔ Learn linear rule in feature space.

Example:



➔ The separating hyperplane in feature space is degree two polynomial in input space.

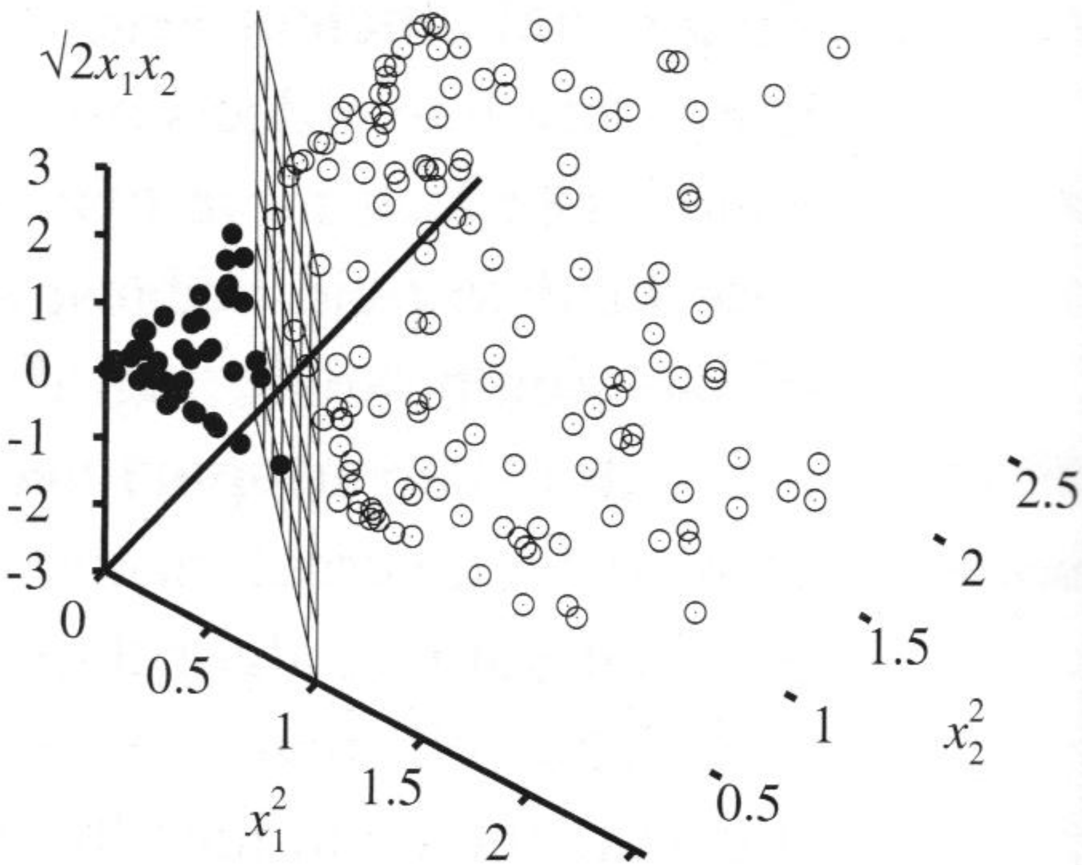


1.5
1
0.5
0
-0.5
-1
-1.5
x_2
-1.5
-1
-0.5
0
0.5
1
1.5
x_1

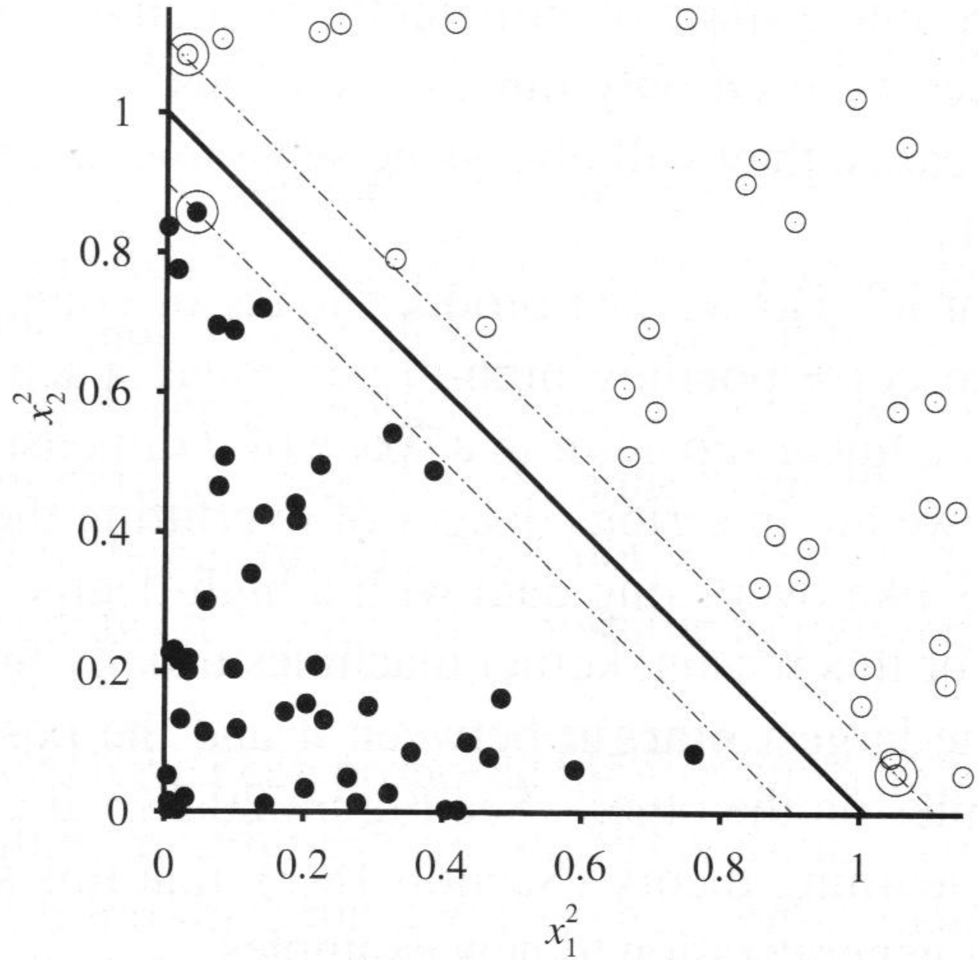
Transformation

- Instead of x_1, x_2 use

$$f_1 = x_1^2 \,, \qquad f_2 = x_2^2 \,, \qquad f_3 = \sqrt{2}x_1x_2$$



$\sqrt{2}x_1x_2$
3
2
1
0
-1
-2
-3
0
0.5
1
1.5
2
x_1^2
0.5
1
1.5
2
2.5
x_2^2



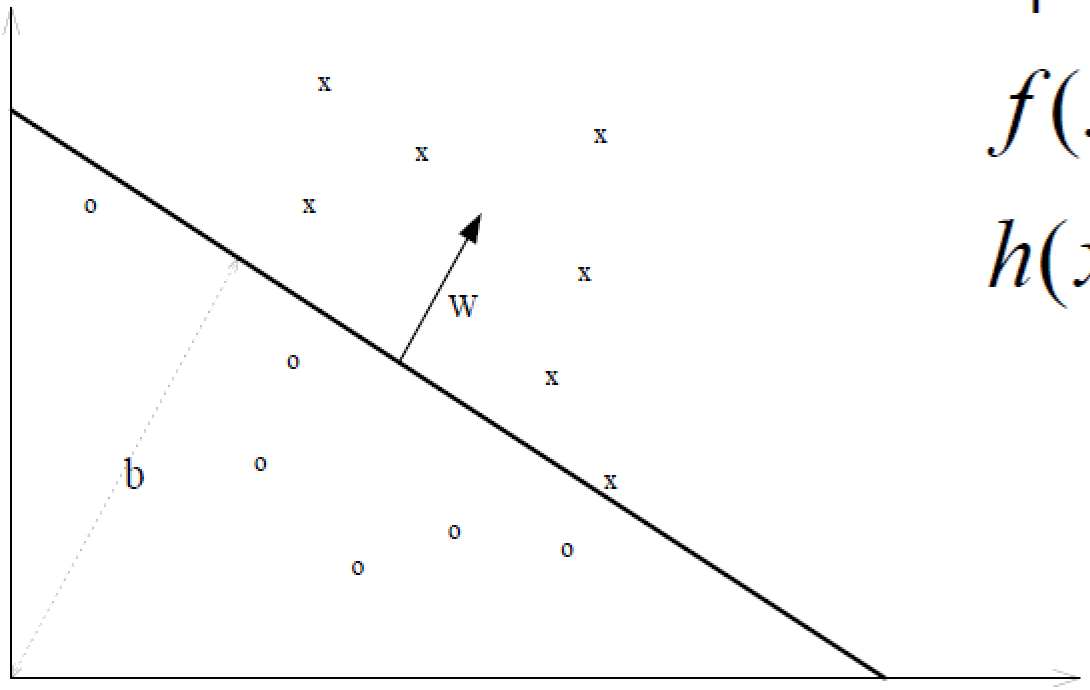
x_2^2
0
0.2
0.4
0.6
0.8
1
0
0.2
0.4
0.6
0.8
1
x_1^2

How do we find these features?

- F(x) $=$

$$f_1 = x_1^2\,, \qquad f_2 = x_2^2\,, \qquad f_3 = \sqrt{2}x_1x_2$$

Reminder



- Linear Separation of the input space

$$f(x) = \langle w, x \rangle + b$$

$$h(x) = sign(f(x))$$

From http://www.support-vector.net/icml-tutorial.pdf

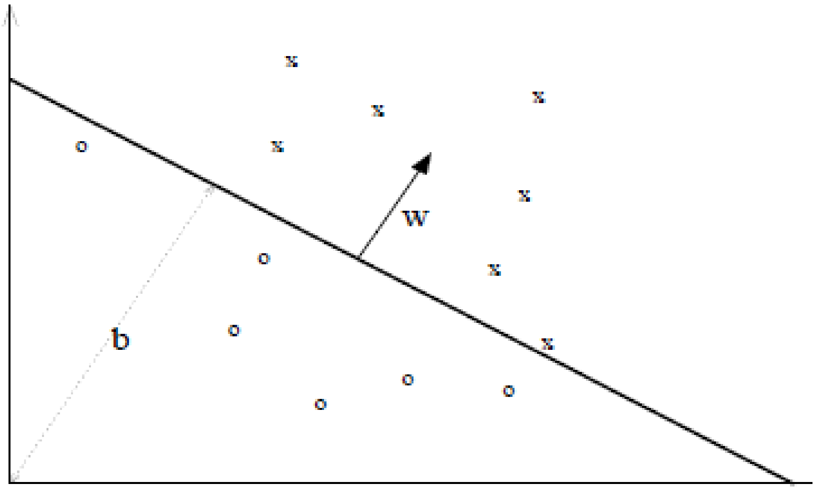
Reminder

Update rule
(ignoring threshold):

- if $y_i(\langle w_k, x_i \rangle) \leq 0$ then

$$w_{k+1} \leftarrow w_k + \eta y_i x_i$$

$$k \leftarrow k + 1$$



From http://www.support-vector.net/icml-tutorial.pdf

Observation

- Solution is a linear combination of training points

$$w = \sum \alpha_i y_i x_i$$

$$\alpha_i \geq 0$$

- Only used informative points (mistake driven)
- The coefficient of a point in combination reflects its 'difficulty'

From http://www.support-vector.net/icml-tutorial.pdf

Dual Representation

possible to rewrite the algorithm using this alternative representation

The decision function can be re-written as follows:

$$f(x) = \langle w, x \rangle + b = \sum \alpha_i y_i \langle x_i, x \rangle + b$$

$$w = \sum \alpha_i y_i x_i$$

From http://www.support-vector.net/icml-tutorial.pdf

New Update Rule

- The update rule can be written as

$$y_i\left(\sum \alpha_j y_j \langle x_j, x_i \rangle + b\right) \leq 0 \quad \text{then } \alpha_i \leftarrow \alpha_i + \eta$$

- And the hypothesis h(x) is

$$h(\mathbf{x}) = \text{sign}\left(\sum_i \alpha_i y_i (\mathbf{x} \cdot \mathbf{x}_i)\right)$$

  – Note: Hypothesis uses only dot products with key examples ("support vectors")

From http://www.support-vector.net/icml-tutorial.pdf

Max Margin = Minimal Norm

- If we fix the functional margin to 1, the geometric margin equal *1/||w||*
- Hence, maximize the margin by minimizing the norm
  - Minimize $\langle w, w \rangle$
  - Subject to $y_i(\langle w, x_i \rangle + b) \geq 1$

From http://www.support-vector.net/icml-tutorial.pdf

How to find the α's?

- Maximize: $$W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$
- Subject to: $\alpha_i \geq 0$

$$\sum_i \alpha_i y_i = 0$$

From http://www.support-vector.net/icml-tutorial.pdf

Using Kernels: Implicit features

- We need to compute $<x_i,x_j>$ many times
  - Or $<F(x_i),F(x_j)>$ if we use features F(x)
- But what if we wound a set of features F(x) such that $F(x_i) \cdot F(x_j) = (x_i \cdot x_j)^2$?
  - Then we only need to compute $(x_i \cdot x_j)^2$
  - We don't even need to know what F is (!)

Implicit features: Example

$$x = (x_1, x_2);$$

$$z = (z_1, z_2);$$

$$\langle x, z \rangle^2 = (x_1 z_1 + x_2 z_2)^2 =$$

$$= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 z_1 x_2 z_2 =$$

$$= \left\langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2), (z_1^2, z_2^2, \sqrt{2} z_1 z_2) \right\rangle =$$

$$= \left\langle \phi(x), \phi(z) \right\rangle$$

www.support-vector.net

Calculate using a Kernel

- Two vectors
  - A = (1,2)
  - B = (3,4)
- Three Features:
  - $F(X) = \{x_1^2,\ x_2^2,\ \sqrt{2} \cdot x_1 \cdot x_2\}$
  - Calculate $F(A) \cdot F(B)$

What is $F(A) \cdot F(B)$?

A=120 B=121 C=144 D=256

Calculate without using a Kernel

- $A = (1,2)$, $B = (3,4)$
- $F(X) = \{x_1^2,\ x_2^2,\ \sqrt{2} \cdot x_1 \cdot x_2\}$
  - $A=(1,2) \rightarrow F(A)=\{1^2, 2^2, \sqrt{2} \cdot 1 \cdot 2\} = \{1, 4, 2\sqrt{2}\}$
  - $B=(3,4) \rightarrow F(B)=\{3^2, 4^2, \sqrt{2} \cdot 3 \cdot 4\} = \{9, 16, 12\sqrt{2}\}$
- $F(A) \cdot F(B) = 1 \cdot 9 + 4 \cdot 16 + 2 \cdot 12 \cdot 2 = 121$

Calculate using a Kernel

$$\langle x, z \rangle^2 = (x_1 z_1 + x_2 z_2)^2 =$$

$$= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 z_1 x_2 z_2 =$$

$$= \left\langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2), (z_1^2, z_2^2, \sqrt{2} z_1 z_2) \right\rangle =$$

- A = (1,2), B = (3,4), F(X) = $\{x_1^2,\ x_2^2,\ \sqrt{2} \cdot x_1 \cdot x_2\}$
- F(A)·F(B) = $(A \cdot B)^2 = (1 \cdot 3 + 2 \cdot 4)^2 = 11^2 = 121$
- We didn't need to explicitly calculate or even know about the terms in F at all!
  - just that F(A)·F(B) = $(A \cdot B)^2$

SVM with Kernel

Training:

maximize: $D(\vec{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j K(\vec{x}_i, \vec{x}_j)$

subject to: $\sum_{i=1}^{n} y_i \alpha_i = 0$

$\forall_{i=1}^{n} : 0 \leq \alpha_i \leq C$

Classification:

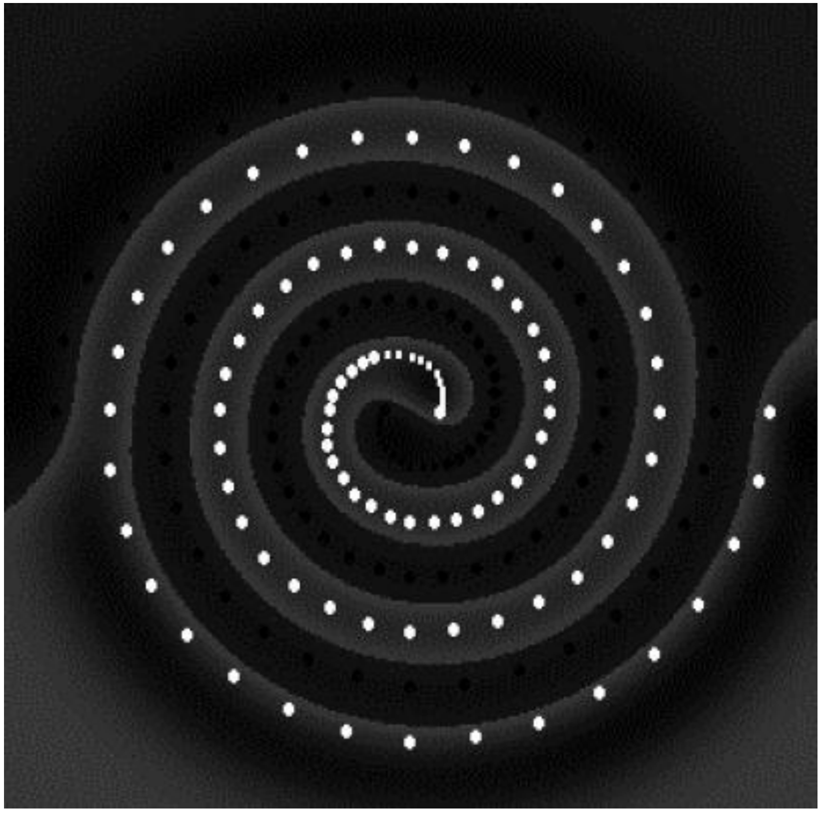
$$h(\vec{x}) = sign\left(\left[\sum_{i=1}^{n} \alpha_i y_i \Phi(\vec{x}_i)\right] \cdot \Phi(\vec{x}) + b\right)$$

$$= sign\left(\sum_{i=1}^{n} \alpha_i y_i K(\vec{x}_i, \vec{x}) + b\right)$$

New hypotheses spaces through new Kernels:

- Linear: $K(\vec{a}, \vec{b}) = \vec{a} \cdot \vec{b}$
- Polynomial: $K(\vec{a}, \vec{b}) = [\vec{a} \cdot \vec{b} + 1]^d$
- Radial Basis Function: $K(\vec{a}, \vec{b}) = exp(-\gamma[\vec{a} - \vec{b}]^2)$
- Sigmoid: $K(\vec{a}, \vec{b}) = tanh(\vec{a} \cdot \vec{b})$

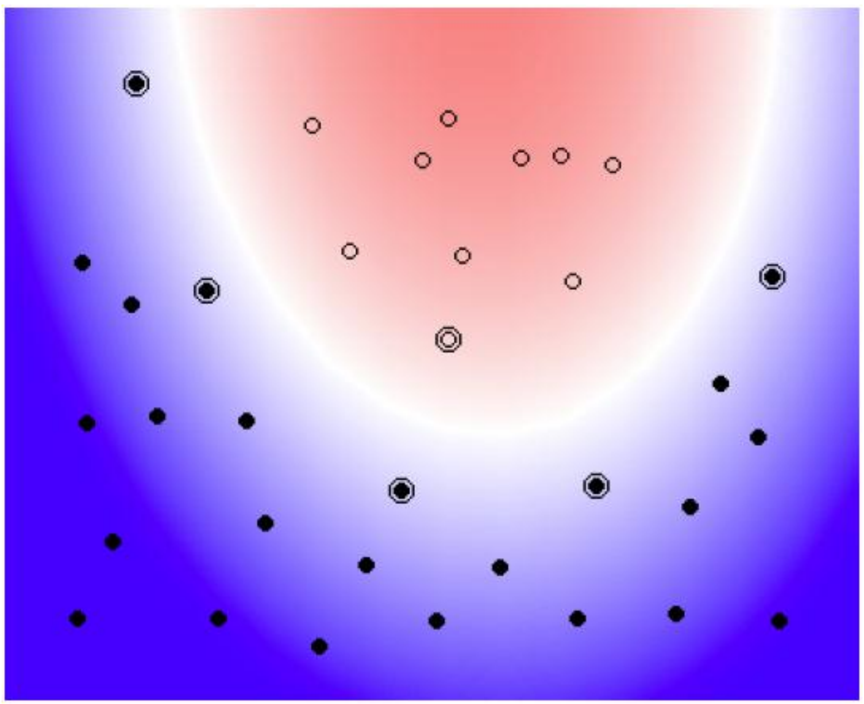
Solution with Gaussian Kernels



Examples of Kernels

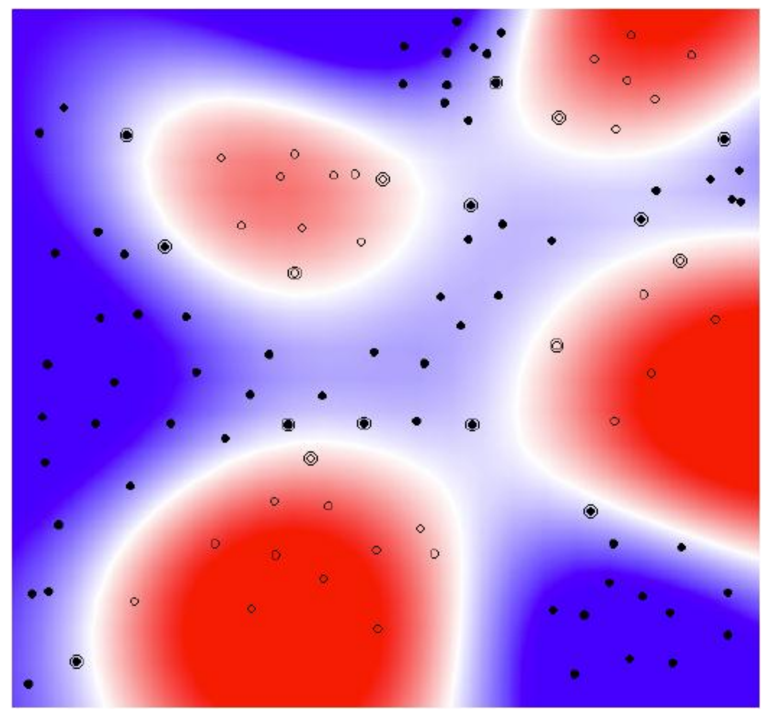
Polynomial Function

$$K(\vec{a},\vec{b}) = [\vec{a}\cdot\vec{b}+1]^2$$

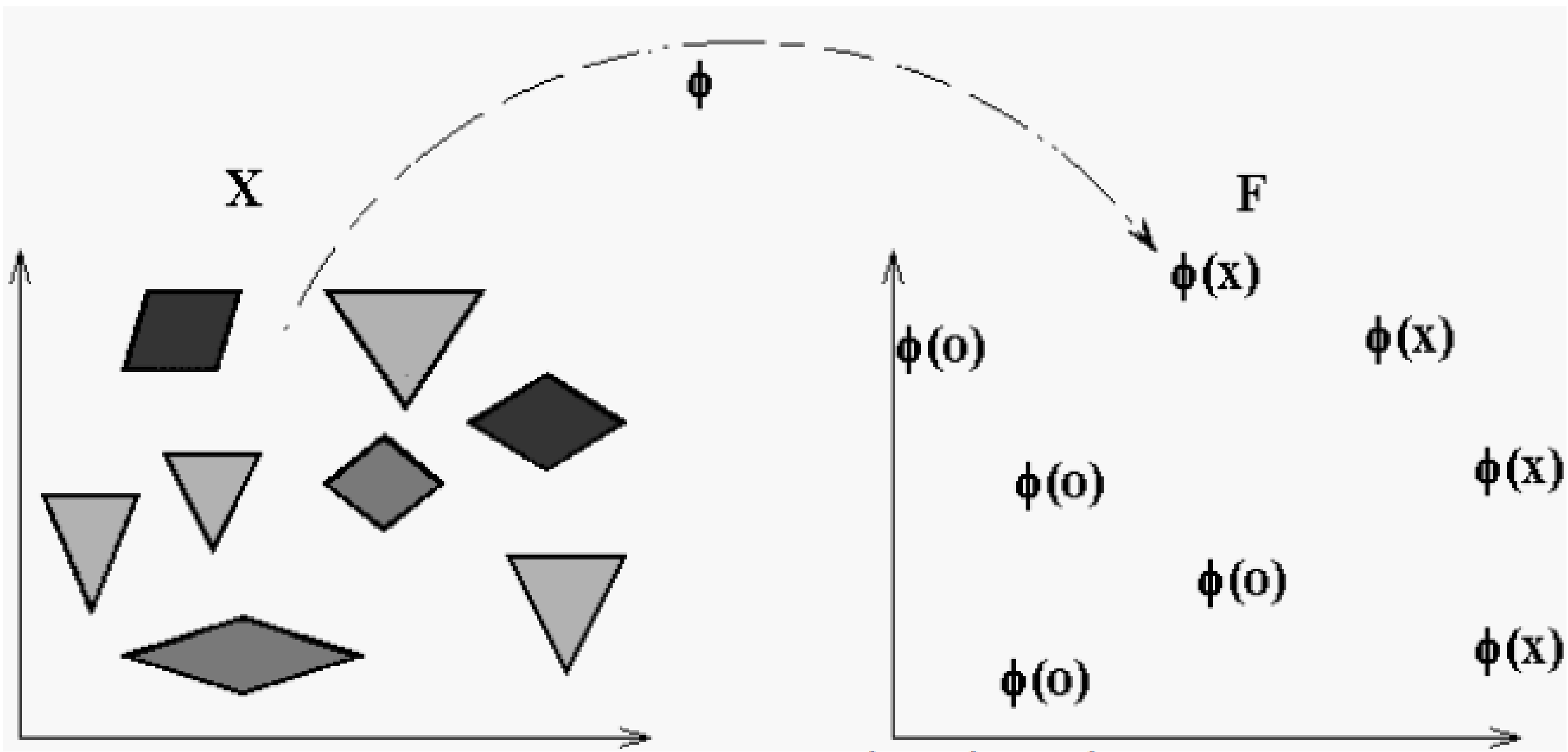


Radial Basis

$$K(\vec{a},\vec{b}) = exp(-\gamma[\vec{a}-\vec{b}]^2)$$



Kernels for Non-Vectorial Data



www.support-vector.net

Kernels for Non-Vectorial Data

- Applications with Non-Vectorial Input Data
  → classify non-vectorial objects
  - Protein classification (x is string of amino acids)
  - Drug activity prediction (x is molecule structure)
  - Information extraction (x is sentence of words)
  - Etc.
- Applications with Non-Vectorial Output Data
  → predict non-vectorial objects
  - Natural Language Parsing (y is parse tree)
  - Noun-Phrase Co-reference Resolution (y is clustering)
  - Search engines (y is ranking)

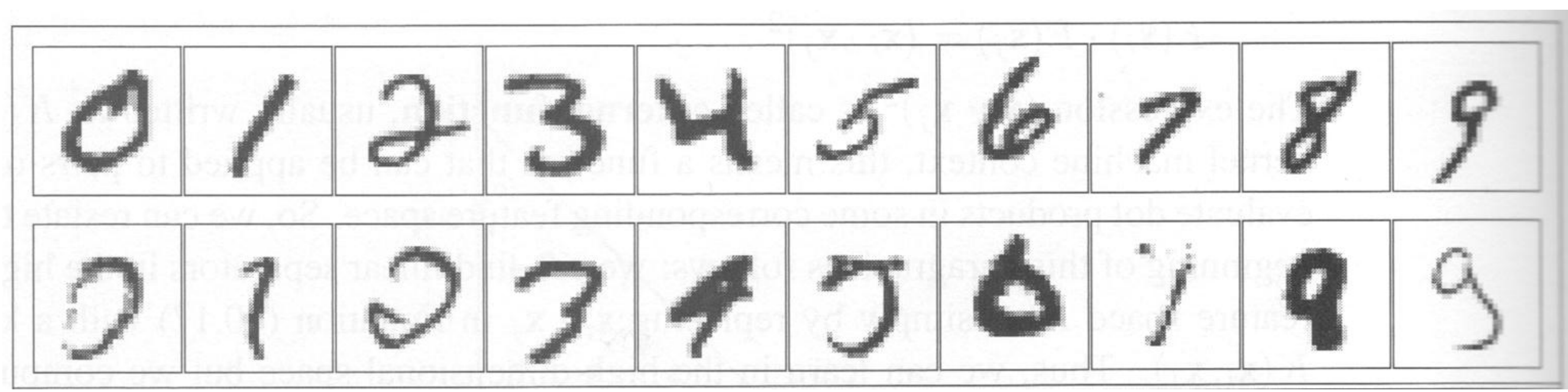
➔ Kernels can compute inner products efficiently!

Properties of SVMs with Kernels

- Expressiveness
  - Can represent any boolean function (for appropriate choice of kernel)
  - Can represent any sufficiently “smooth” function to arbitrary accuracy (for appropriate choice of kernel)
- Computational
  - Objective function has no local optima (only one global)
  - Independent of dimensionality of feature space
- Design decisions
  - Kernel type and parameters
  - Value of C

Benchmark

- Character recognition
- NIST Database of handwritten digits
  - 60,000 samples
  - 20x20 pixels



0000067894 12345678 0145 0000029545

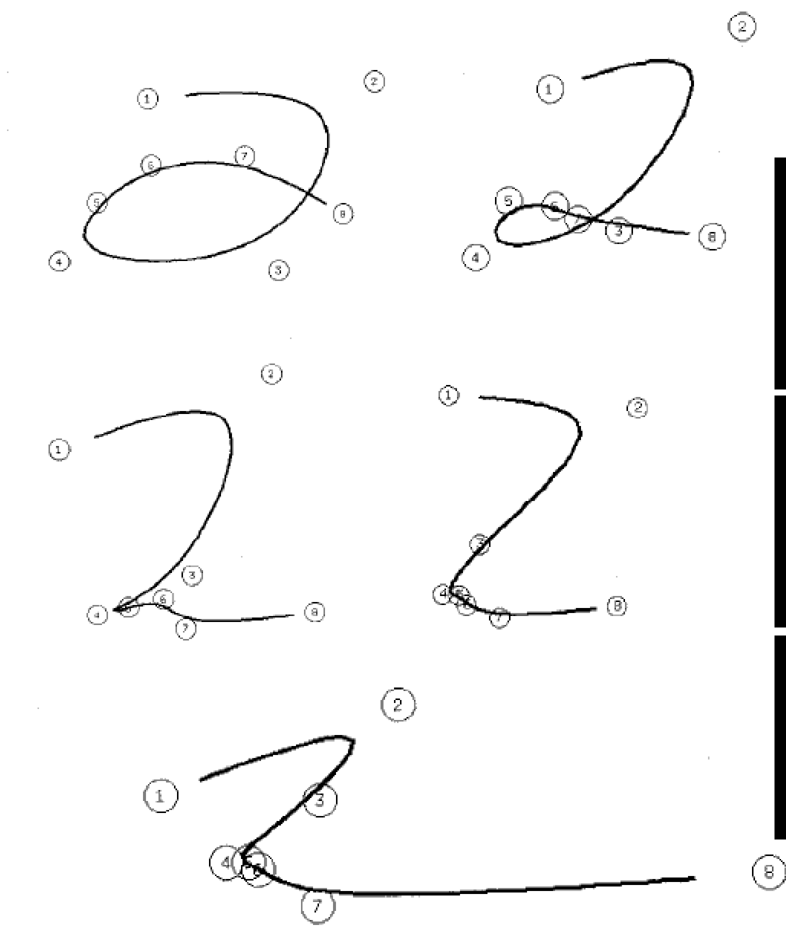
Digit Recognition

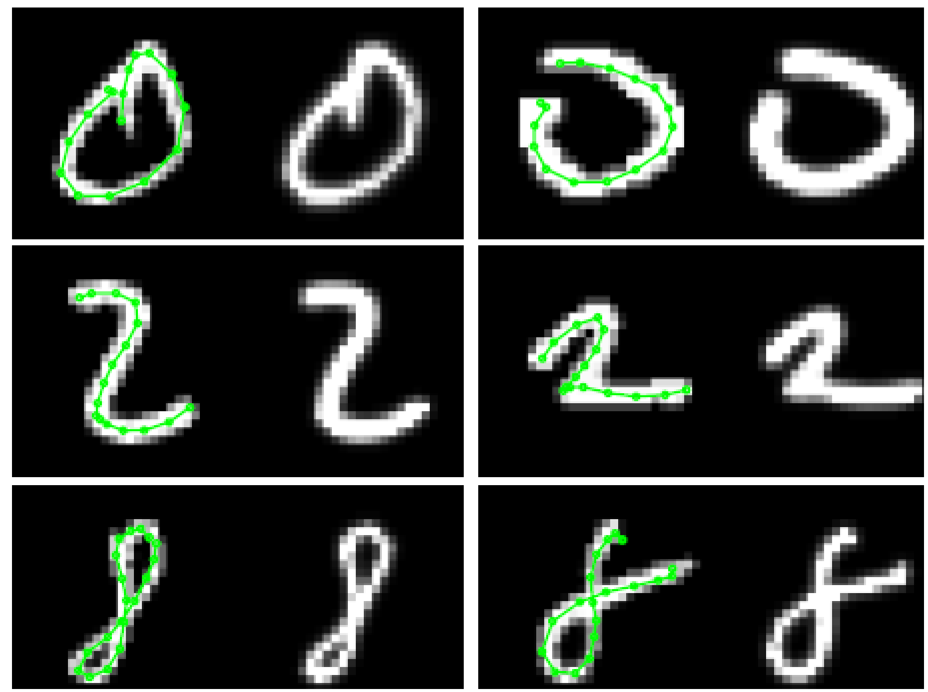
- 3-nearest neighbor: 2.4% error rate
  - stores all samples
- Single hidden layer NN: 1.6%
  - 400 inputs, 10 output, 300 hidden (using CV)
- Specialized nets (LeNet): 0.9%
  - Use specialized 2D architecture
- Boosted NN: 0.7%
  - Three copies of LeNets

Digit Recognition

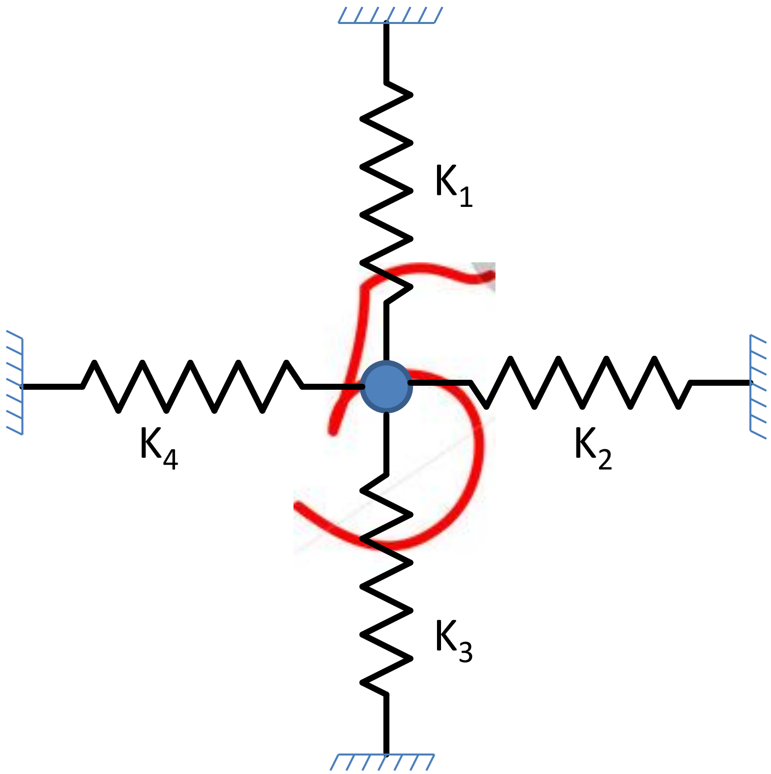
- SVM: 1.1%
  - Compare to specialized LeNet 0.9%
- Specialized SVM: 0.56%
- Shape Matching: 0.63%
  - Machine vision techniques
- Humans?
  - A=0.1% B=0.5% C=1% D=2.5% E=5%

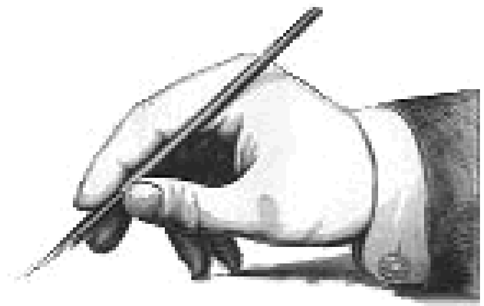
Generative models

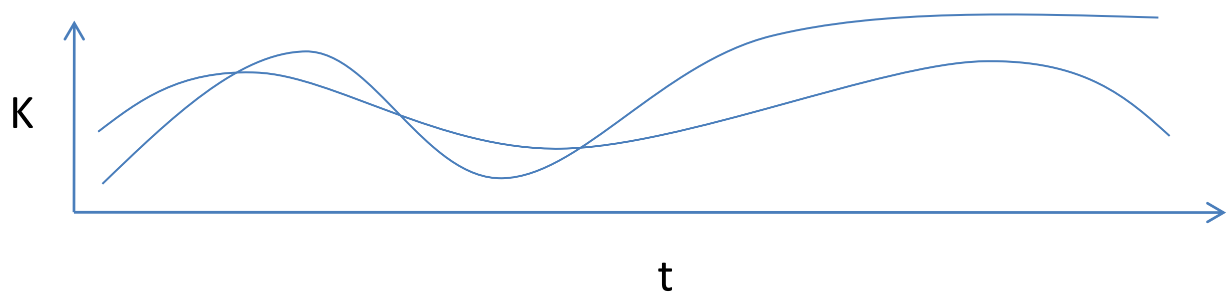




Hinton et al


K1
K2
K3
K4




K
t