Support Vector Machines (SVM)

- a new classifier
- Attractive because
  - Has sound mathematical foundations
  - Performs very well in diverse and difficult applications
  - See paper placed on the class website

Review of basic analytical geometry

- Dot product of vectors by coordinates and with the angle
- If vectors a, b are perpendicular, then $(a \bullet b) = 0$ (e.g. $(0, c) \bullet (d, 0) = 0$
- A hyperplane in an n-dimensional space has the property $\{x | (w \bullet x) + b = 0\}$; w is the *weight vector*, b is the *threshold;* $x = (x_1, \ldots, x_n)$; $w = (w_1, \ldots, w_n)$
- A hyperplane divides the n-dimensional space into two subspaces: one is $\{y | y((w \bullet x) + b) > 0\}$, the other is complementary $(y | y((w \bullet x) + b) < 0)$

- Lets revisit the general classification problem.
- We want to estimate an *unknown* function f, all we know about it is the training set $(\mathrm{x}_1, y_1), \ldots (\mathrm{x}_n, y_n)$
- The objective is to minimize the expected error (risk)

  $$R[f] = \int l(f(\mathrm{x}), y)\, dP(\mathrm{x}, y)$$

  where *l* is a *loss function,* eg

  $$l(f(x), y) = \Theta(-yf(x))$$

  and $\Theta(z) = 0$ for $z<0$ and $\Theta(z)=1$ otherwise
- Since we do not know *P*, we cannot measure risk
- We want to approximate the true error (risk) by the empirical error (risk):

  $$R_{emp}[f] = \frac{1}{n}\sum_{i=1}^{n} l(f(\mathrm{x}_i, y_i))$$

- We know from the PAC theory that conditions can be given on the learning task so that the empirical risk converges towards the true risk
- We also know that the difficulty of the learning task depends on the complexity of *f* (VC dimension)
- It is known that the following relationship between the empirical risk and the complexity of the language (*h* denotes VC dimension of the class of *f)* :

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{h(\ln\frac{2n}{h}+1) - \ln(\partial / 4)}{n}}$$

is true with probability at least δ for n> h

SRM

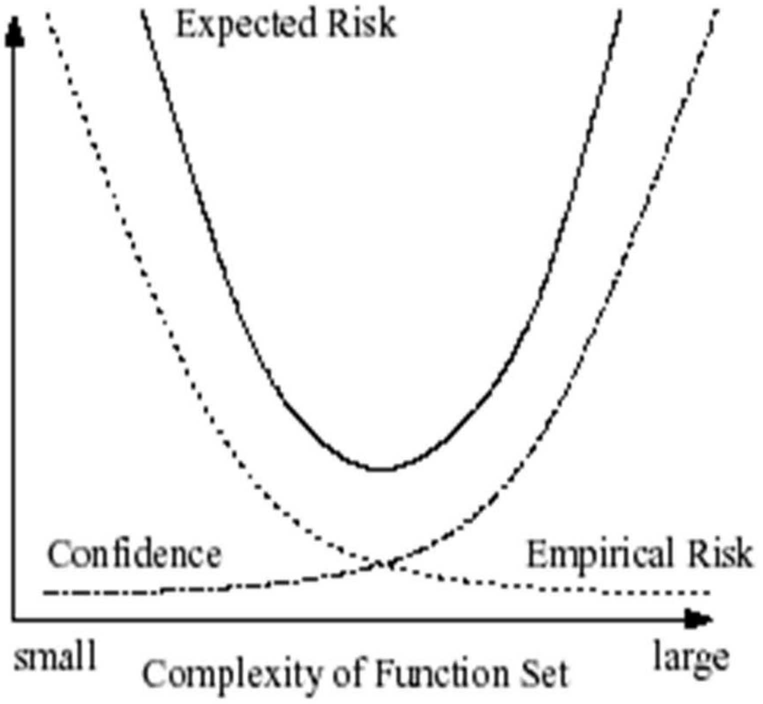


Fig. 2. Schematic illustration of Eq. (3). The dotted line represents the training error (empirical risk), the dash-dotted line the upper bound on the complexity term (confidence). With higher complexity the empirical error decreases but the upper bound on the risk confidence becomes worse. For a certain complexity of the function class the best expected risk (solid line) is obtained. Thus, in practice the goal is to find the best tradeoff between empirical error and complexity.

Structural Risk Minimization (SRM) chooses the class of F to find a balance between the simplicity of f (very simple may result in a large empirical risk) and and the empirical risk (small may require a class function with a large h)

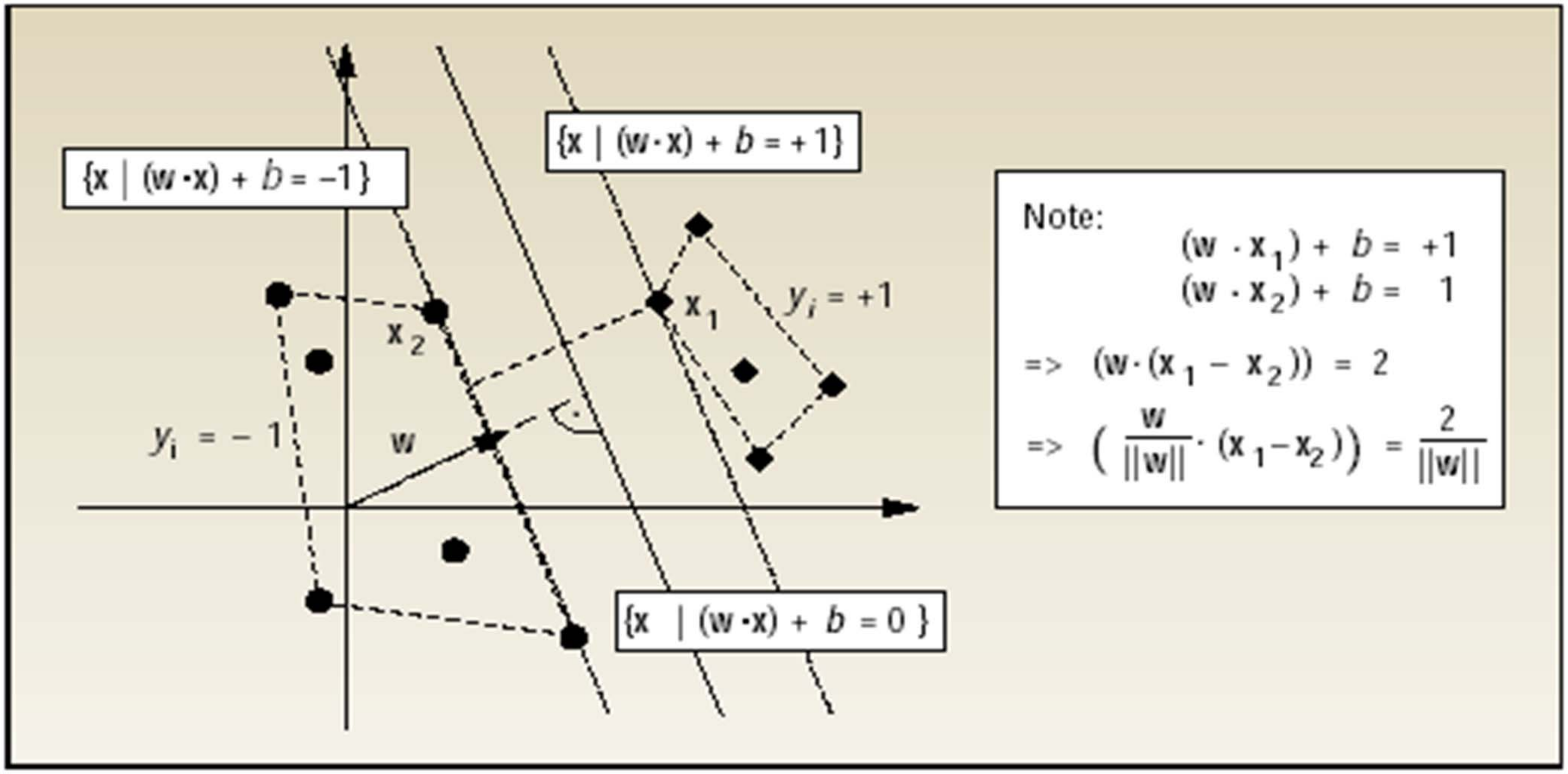
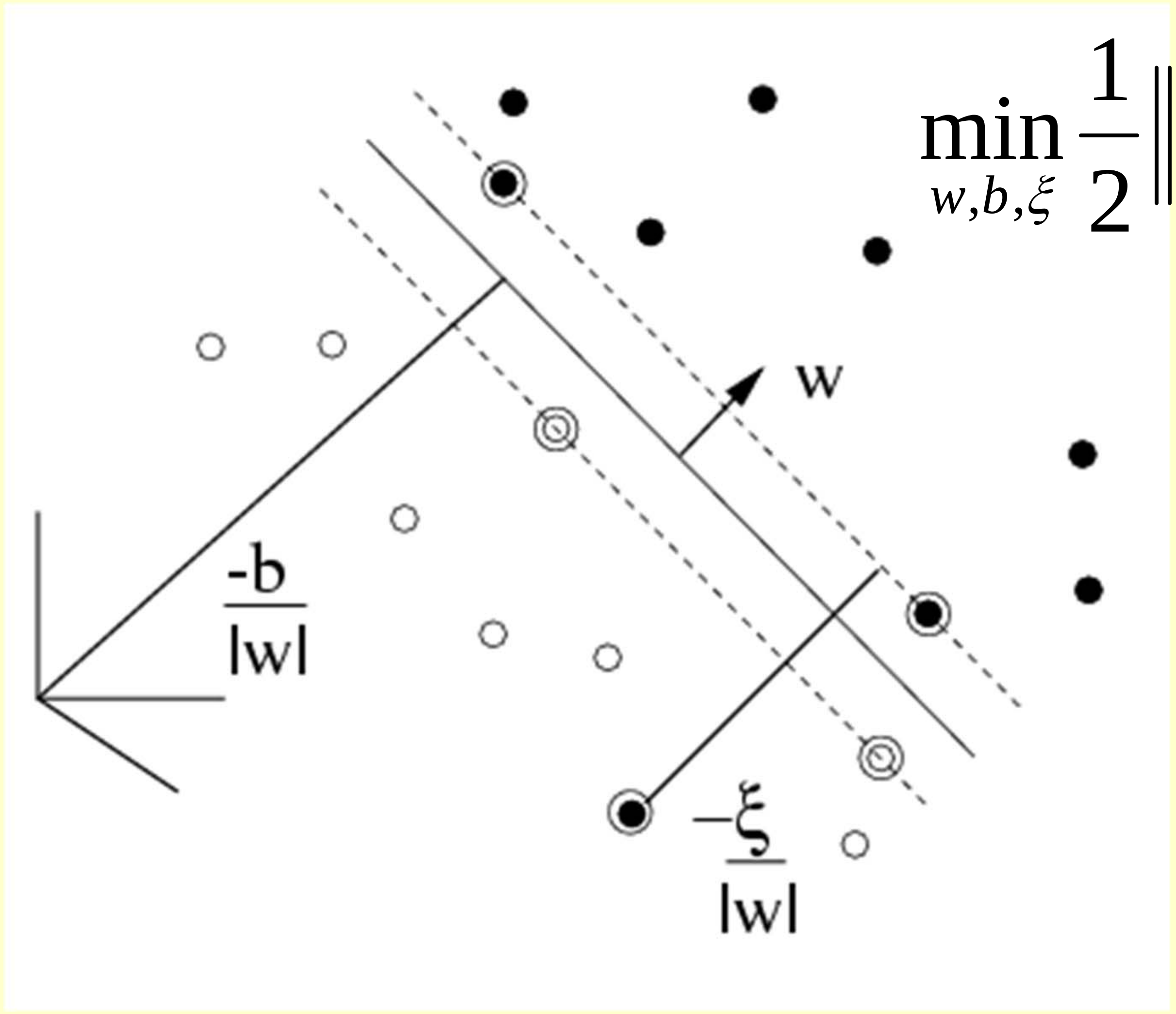


Figure 1. A separable classification toy problem: separate balls from diamonds. The optimal hyperplane is orthogonal to the shortest line connecting the convex hulls of the two classes (dotted), and intersects it half way. There is a weight vector **w** and a threshold b such that $y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) > 0$. Rescaling **w** and b such that the point(s) closest to the hyperplane satisfy $|(\mathbf{w} \cdot \mathbf{x}_i) + b| = 1$, we obtain a form $(\mathbf{w}, b)$ of the hyperplane with $y_i((\mathbf{w} \cdot x_i) + \mathrm{b}) \geq 1$. Note that the margin, measured perpendicularly to the hyperplane, equals $2/\|\mathbf{w}\|$. To maximize the margin, we thus have to minimize $|\mathbf{w}|$ subject to $y_i((\mathbf{w} \cdot \mathbf{x}_i) + \mathrm{b}) \geq 1$.

Points lying on the margin are called *support vectors;*

w can be constructed efficiently – quadratic optimization problem.

- Derivation of the optimal margin (see Ch. 3 of the book “Ten top data mining algorithms)
- “soft margin” idea and slack variable



$$\min_{w,b,\xi} \frac{1}{2}\left\|w^2\right\| + C\sum_{i=1}^{n}\xi_i$$

Basic idea of SVM

- Linearly separable problems are easy (quadratic), but of course most problems are not l. s.
- Take any problem and transform it into a high-dimensional space, so that it becomes linearly separable, but
- Calculations to obtain the separability plane can be done in the original input space *(kernel trick)*

Basic idea of SVM

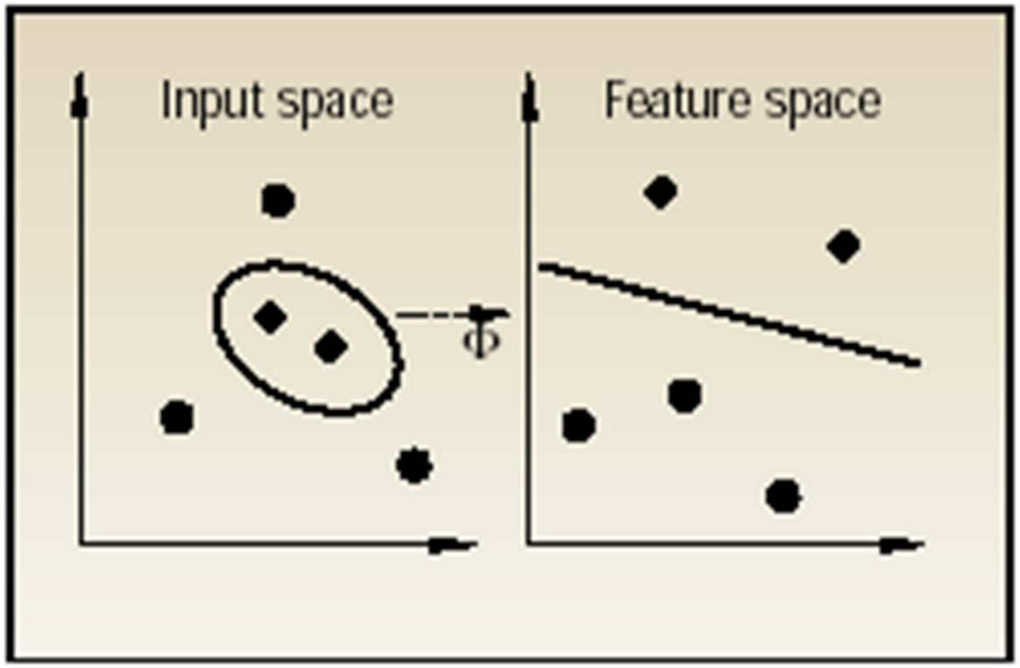


Figure 2. The idea of SV machines: map the training data nonlinearly into a higher-dimensional feature space via Φ, and construct a separating hyperplane with maximum margin there. This yields a nonlinear decision boundary in input space. By the use of a kernel function, it is possible to compute the separating hyperplane without explicitly carrying out the map into the feature space.

Original data is mapped into another dot product space called feature space *F* via a non-linear map Φ:

$$\Phi : R^N \to F$$

Then linear separable classifier is performed in *F*

Note that the only operations in F are dot products*:*

$$k(x, y) = (\Phi(x) \bullet \Phi(y))$$

Consider e.g.

$$\Phi : R^2 \to R^3$$

$$(x_1, x_2) \to (z_1, z_2, z_3) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$$

Lets see that Φ geometrically, and that it does what we want it to do : transform a hard classification problem into an easy one, albeit in a higher dimension

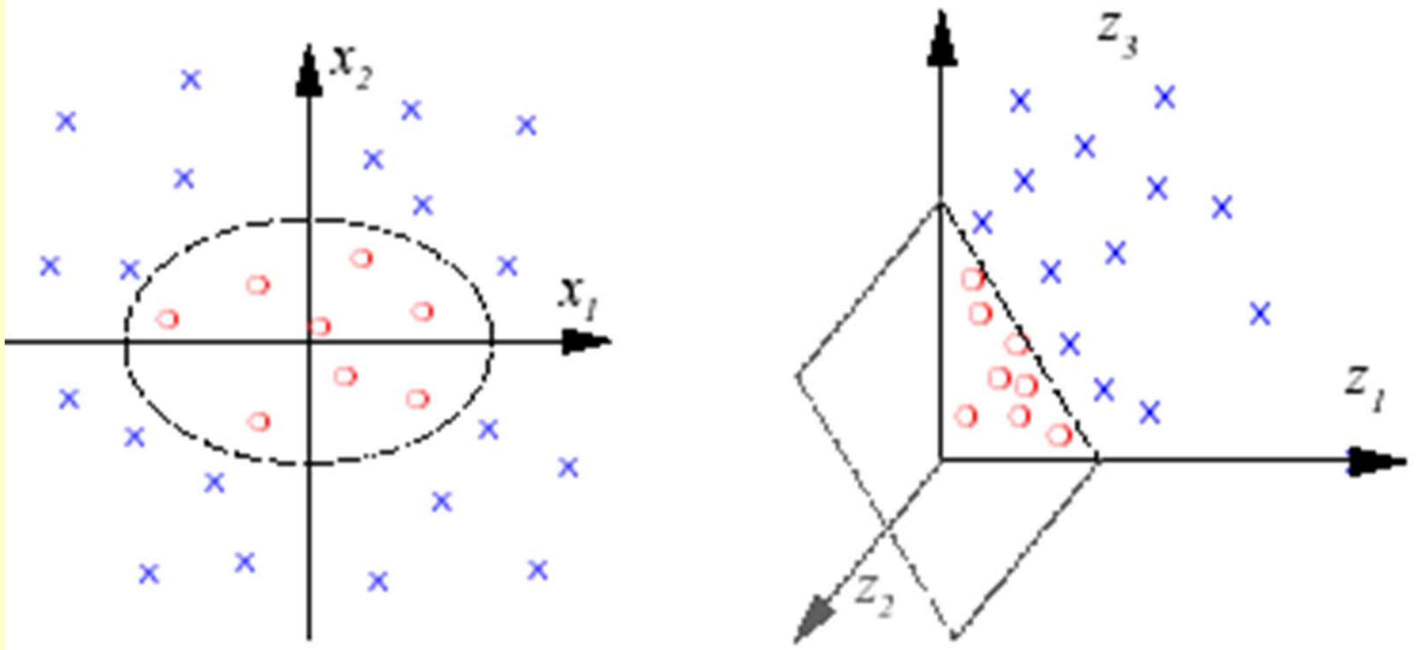


Fig. 4. Two dimensional classification example. Using the second order monomials $x_1^2, \sqrt{2}x_1x_2$ and x_2^2 as features a separation in feature space can be found using a *linear* hyperplane (right). In input space this construction corresponds to a *non-linear* ellipsoidal decision boundary (left) (figure from [48]).

But in general quadratic optimization in the feature space could be very expensive

Consider classifying 16 x 16 pixel pictures, and 5th order monomials

Feature space dimension in this example is $O(\binom{256}{5}) \approx 10^{10}$

Here we show that the transformation from ellipsoidal decision space to a linear one, requiring dot product in the the feature space, can be performed by a kernel function in the input space:

$$\Phi(\mathrm{x}) \bullet \Phi(\mathrm{y}) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)(y_1^2, \sqrt{2} y_1 y_2, y_2^2) =$$

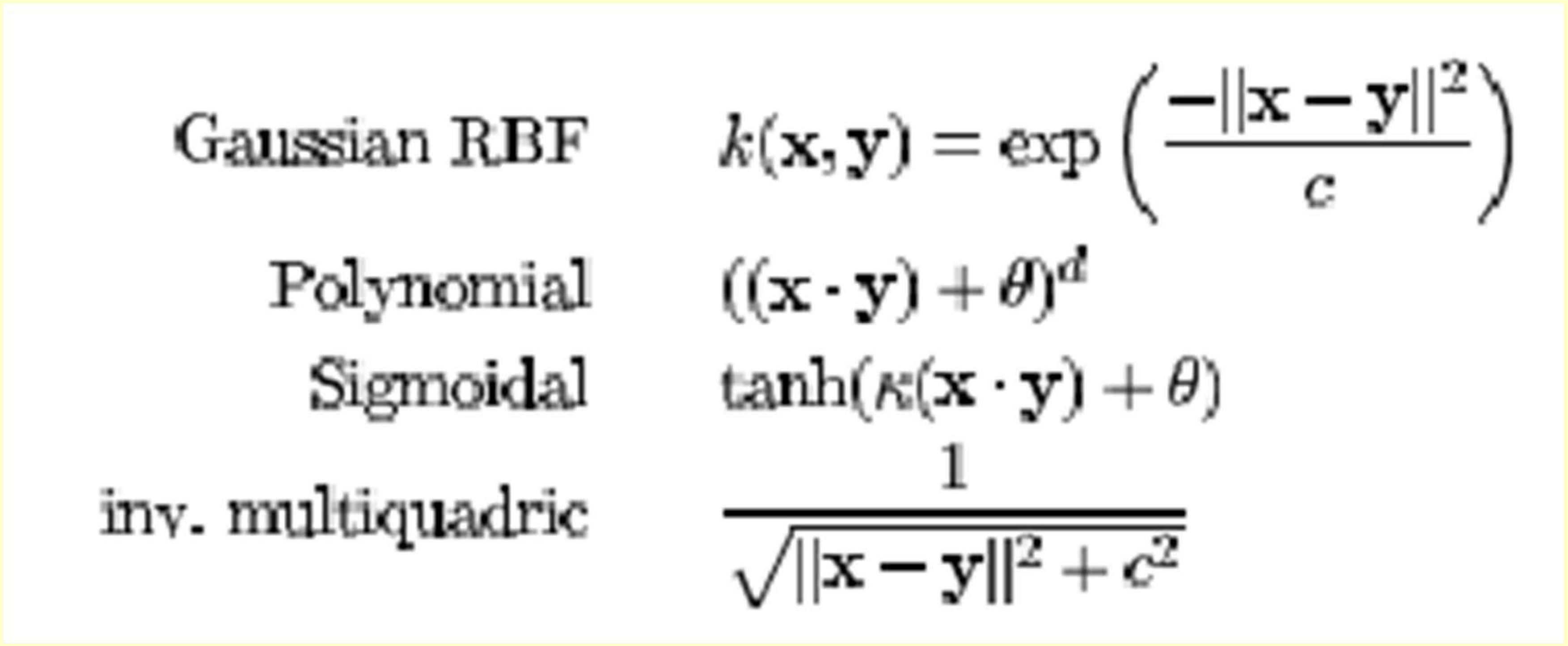
$$((x_1, x_2)(y_1, y_2))^2 = (\mathrm{x} \bullet \mathrm{y})^2 =: k(\mathrm{x}, \mathrm{y})$$

in general, $\mathrm{k}(\mathrm{x},\mathrm{y}) = (\mathrm{x} \bullet \mathrm{y})^{\mathrm{d}}$ computes in the input space

kernels replace computation in FS by computation in the input space

in fact, the transformation Φ needs not to be applied when a kernel is used!

Some common kernels used:



Gaussian RBF	$k(\mathbf{x},\mathbf{y}) = \exp\left(\frac{-\lVert\mathbf{x}-\mathbf{y}\rVert^2}{c}\right)$
Polynomial	$((\mathbf{x}\cdot\mathbf{y})+\theta)^d$
Sigmoidal	$\tanh(\kappa(\mathbf{x}\cdot\mathbf{y})+\theta)$
inv. multiquadric	$\frac{1}{\sqrt{\lVert\mathbf{x}-\mathbf{y}\rVert^2+c^2}}$

Using different kernels we in fact use different classifiers in the input space: gaussian, polynomial, 3-layer neural nets, …

Simplest kernel

- Is the linear kernel $(\mathbf{w} \bullet \mathbf{x}) + b$
- But this only works if the training set is linearly separable. This may not be the case
  - For the linear kernel, or even
  - In the feature space

Classification with SVMs:

- Convert each example x to $\Phi(x)$
- Perform optimal hyperplane algorithm in F; but since we use the kernel all we need to do is to compute

$$
\begin{aligned}
f(\mathbf{x}) &= \operatorname{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i)) + b\right) \\
&= \operatorname{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i \, \mathrm{k}(\mathbf{x}, \mathbf{x}_i) + b\right).
\end{aligned}
$$

where x_i, y_i are training instances, α_i are computed as the solution to the quadratic programming problem

Examples of classifiers in the input space

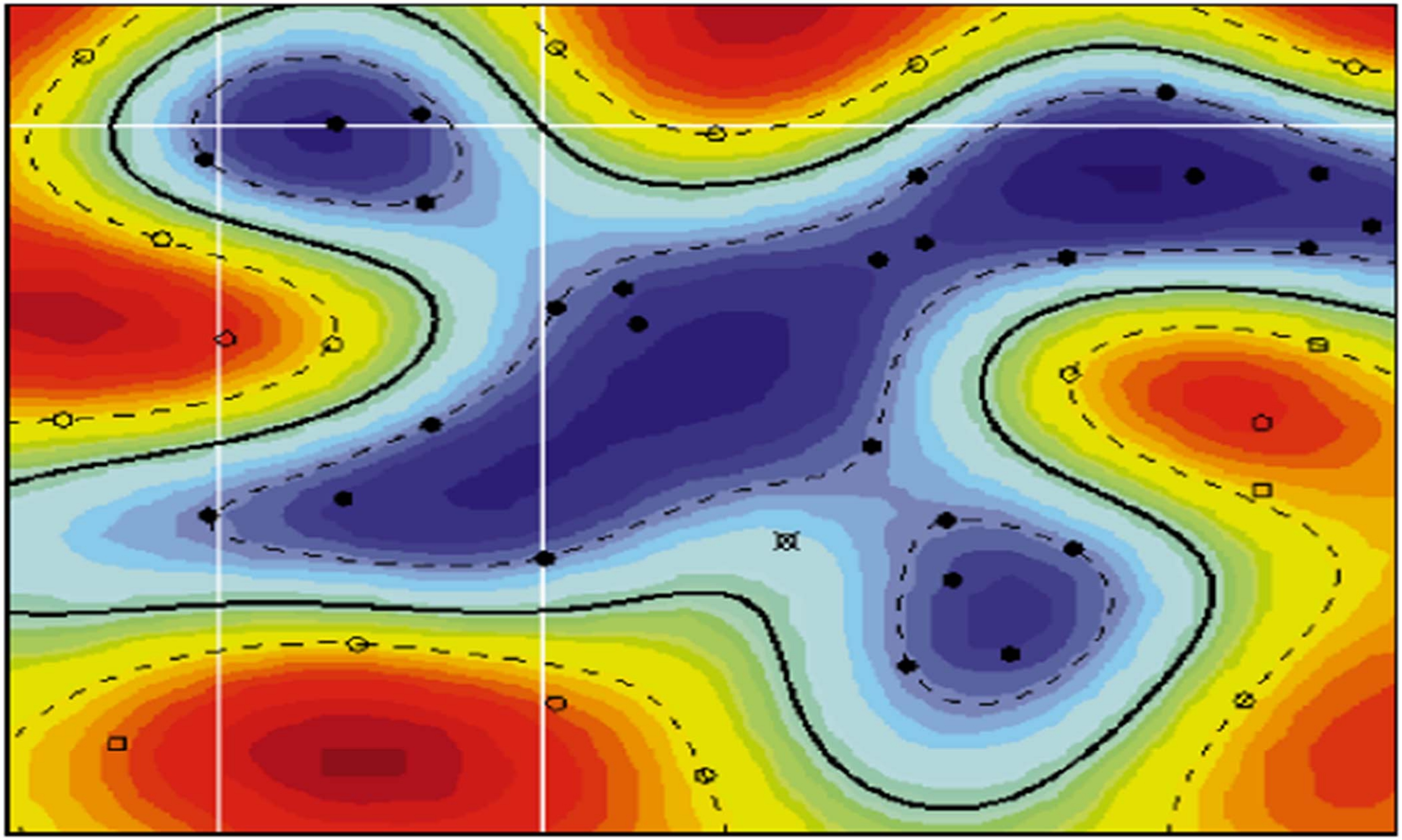


Figure 3. Example of an SV classifier found by using a radial basis function kernel (Equation 8). Circles and disks are two classes of training examples; the solid line is the decision surface; the support vectors found by the algorithm lie on, or between, the dashed lines. Colors code the modulus of the argument $\sum_i v_i \cdot k(\mathbf{x}, \mathbf{x}_i) + b$ of the decision function in Equation 10.

Geometric interpretation of SVM classifier

- Normalize the weight vector to 1 ($\|w\|_2 = 1$) and set the threshold b = 0
- The set of all w that separate training set is

$$\mathcal{V} = \{\mathbf{w} \mid y_i f(\mathbf{x}_i) > 0; i = 1, \cdots, n, \|\mathbf{w}\|_2 = 1\}$$

- But this is the Version Space
- VS has a geometric centre (Bayes Optimal Classifier) near the gravity point
- If VS has a shape in which SVM solution is far from the VS centre, SVM works poorly

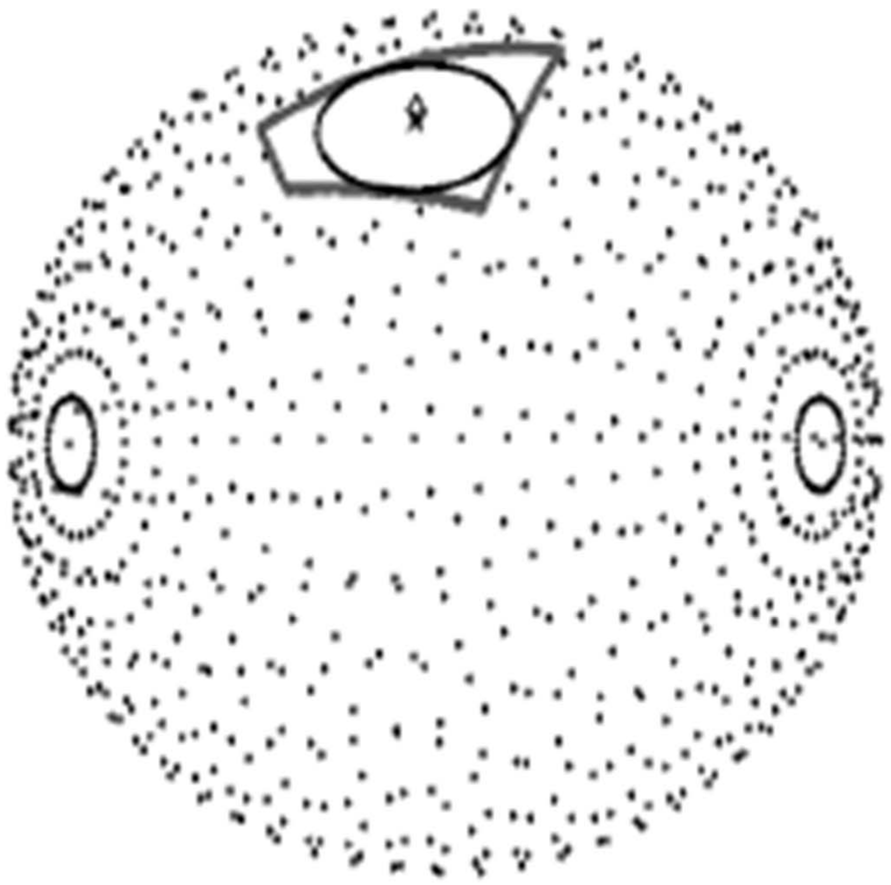
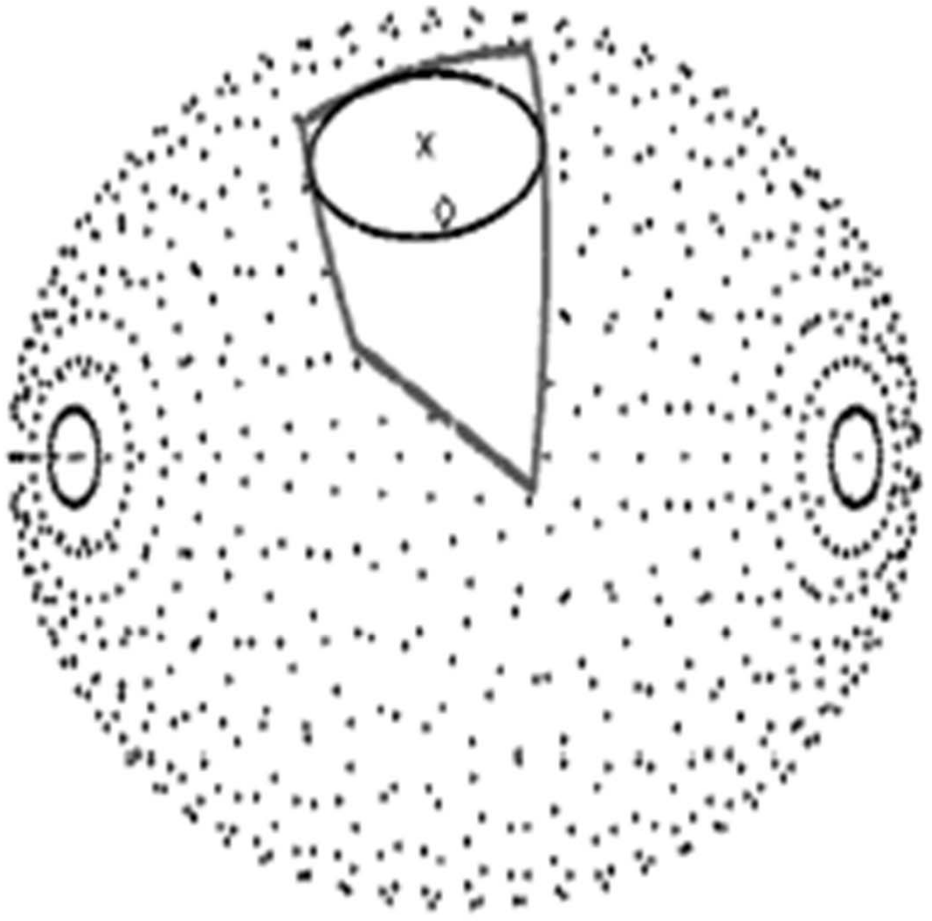


Fig. 5. An example of the version space where the SVM works fine. The center of mass (◇) is close to the SVM solution (×). Figure taken from [72].



ig. 6. An example of the version space where SVM works poorly. The version space has an elongated shape and the center of mass (◇) is far from the SVM solution (×). Figure taken from [72].

Applications

- Text classification
- Image analysis – face recognition
- Bioinformatics – gene expression
- Can the kernel reflect domain knowledge?

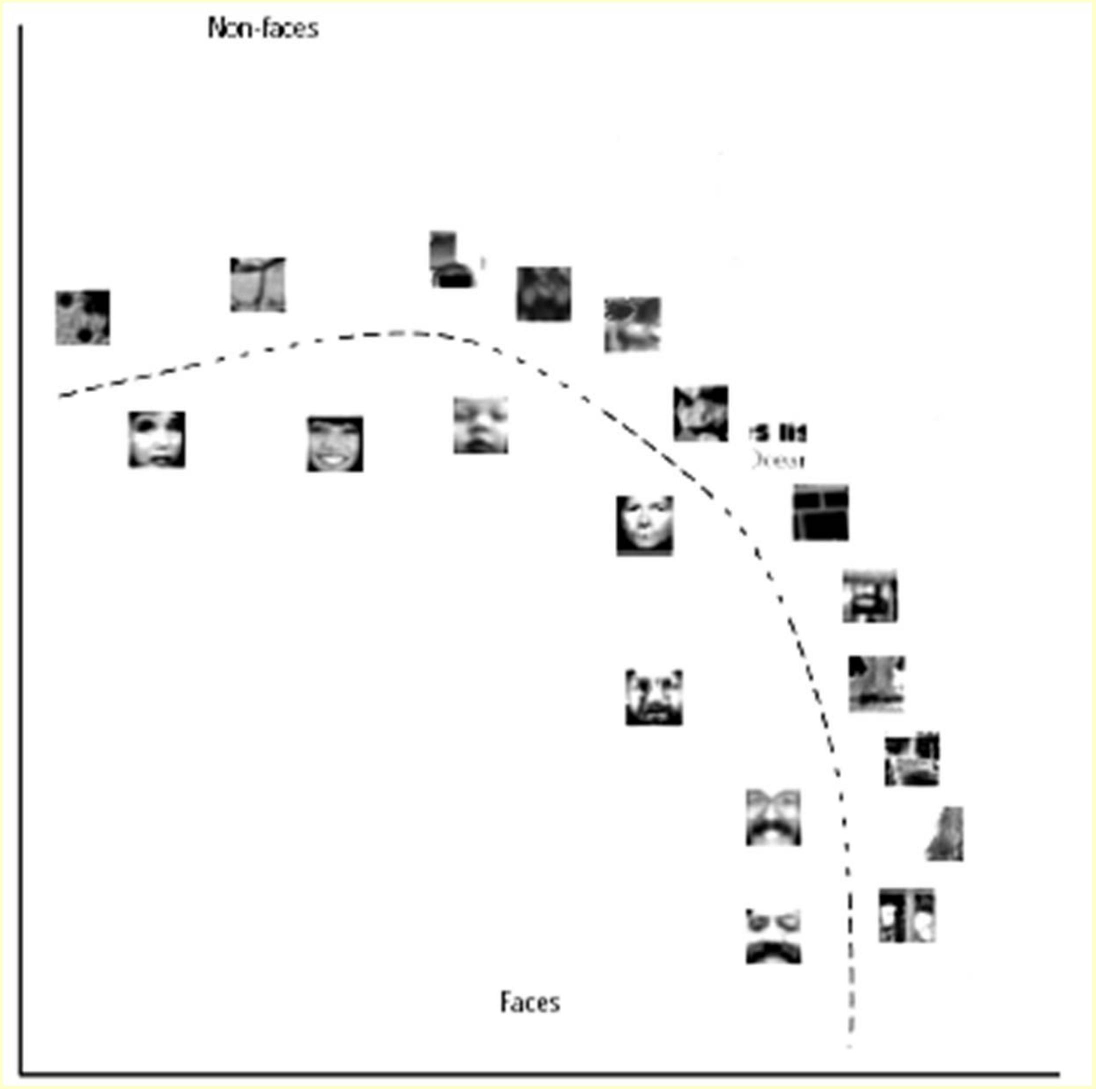
SVM cont'd

- A method of choice when examples are represented by vectors or matrices
- Input space cannot be readily used as attribute-vector (e.g. too many attrs)
- Kernel methods: map data from input space to feature space (FS); perform learning in FS provided that examples are only used within dot point (the *kernel trick* – $\varphi(\mathrm{x}) \bullet \varphi(\mathrm{x}') = k(\mathrm{x}, \mathrm{x}')$)
- SVM but also Perceptron, PCA, NN can be done on that basis
- Collectively – ***kernel-based methods***
- The kernel defines the classifier
- The classifier is independent of the dimensionality of the FS – can even be infinite (gaussian kernel)
- LIMITATION of SVMs: they only work for two-class problems
- Remedy: use of ECOC

Applications – face detection [IEEE INTELLIGENT SYSTEMS]

- The task: to find a rectangle containing a face in an image applicable in face recognition, surveillance, HCI etc. Also in medical image processing and structural defects
- Difficult task – variations that are hard to represent explicitly (hair, moustache, glasses)
- Cast as a classification problem: image regions that are faces and non-faces
- Scanning the image in multiple scales, dividing it into (overlapping) frames and classifying the frames with an SVM:

Face detection cont'd



SVM performing face detection –support vectors are faces and non-faces

Examples are 19x19 pixels, class +1 or -1

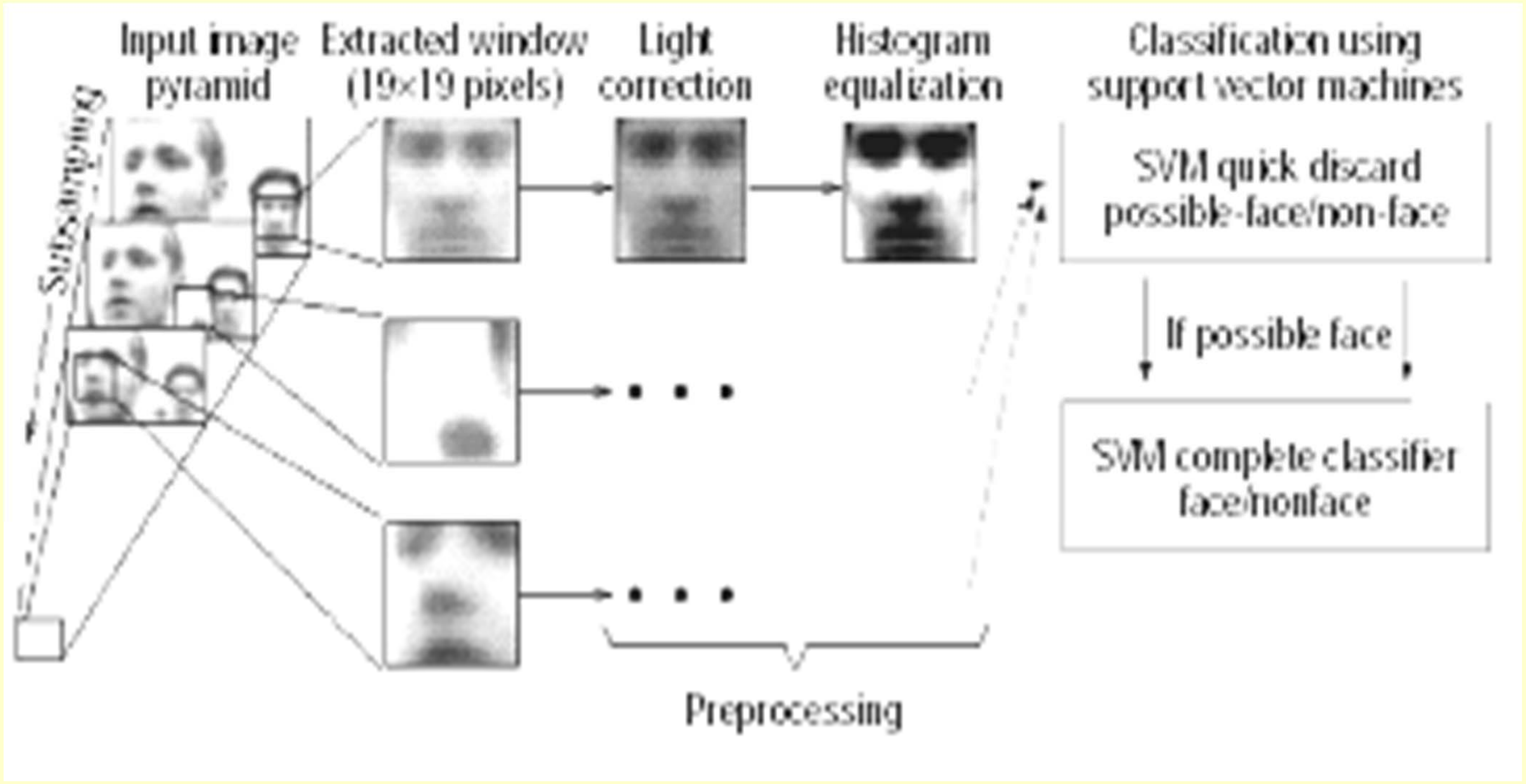
SVM: 2nd degree polynomial with slack variables

Representation tricks: masking out near-boundary area - 361->283, removes noise

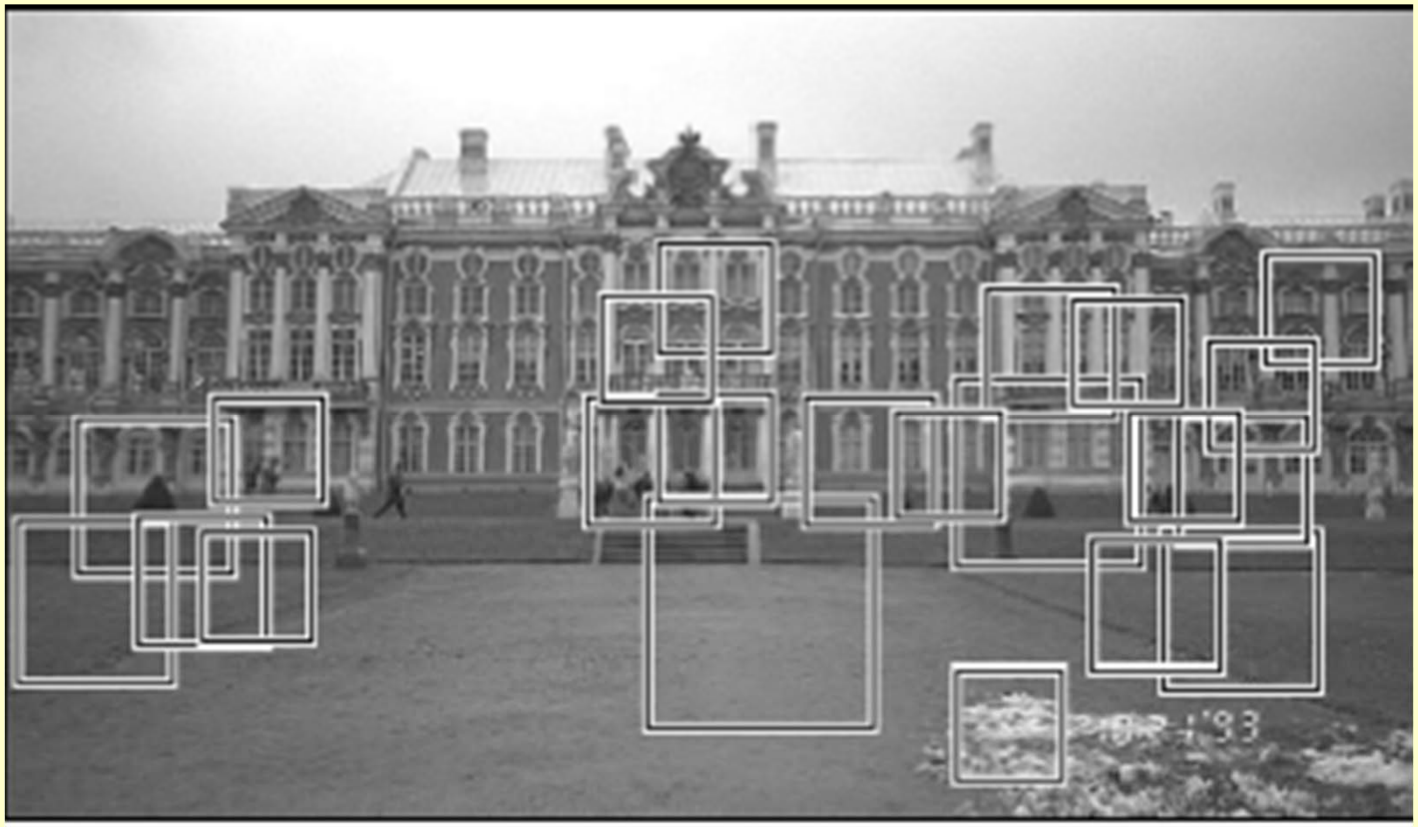
illumination correction: reduction of light and shadow

Discretization of pixel brightness by histogram equalization

Face detection – system architecture


Input image pyramid
Subsampling
Extracted window (19×19 pixels)
Light correction
Histogram equalization
Classification using support vector machines
SVM quick discard possible-face/non-face
If possible face
SVM complete classifier face/nonface
Preprocessing

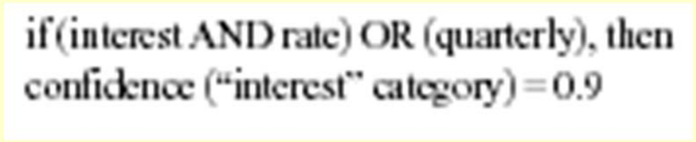
Bootstrapping: using the system on images with no faces and storing false positives to use as negative examples in later training



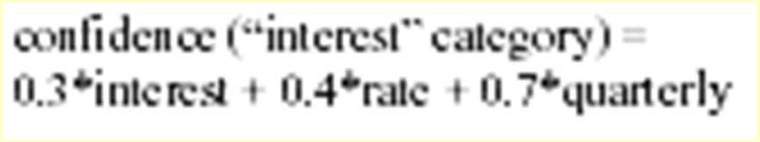
Performance on 2 test sets:
Set A = 313 high quality
Images with 313 faces, set B=
23 images with 155 faces
This results in >4M frames
for A and >5M frames for B.
SVM achieved recall of 97%
on A and 74% on B, with
4 and 20 false positives, resp.

SVM in text classification

- Example of classifiers (the Reuters corpus – 13K stories, 118 categories, time split)
- Essential in document organization (emails!), indexing etc.



if (interest AND rate) OR (quarterly), then confidence ("interest" category) = 0.9



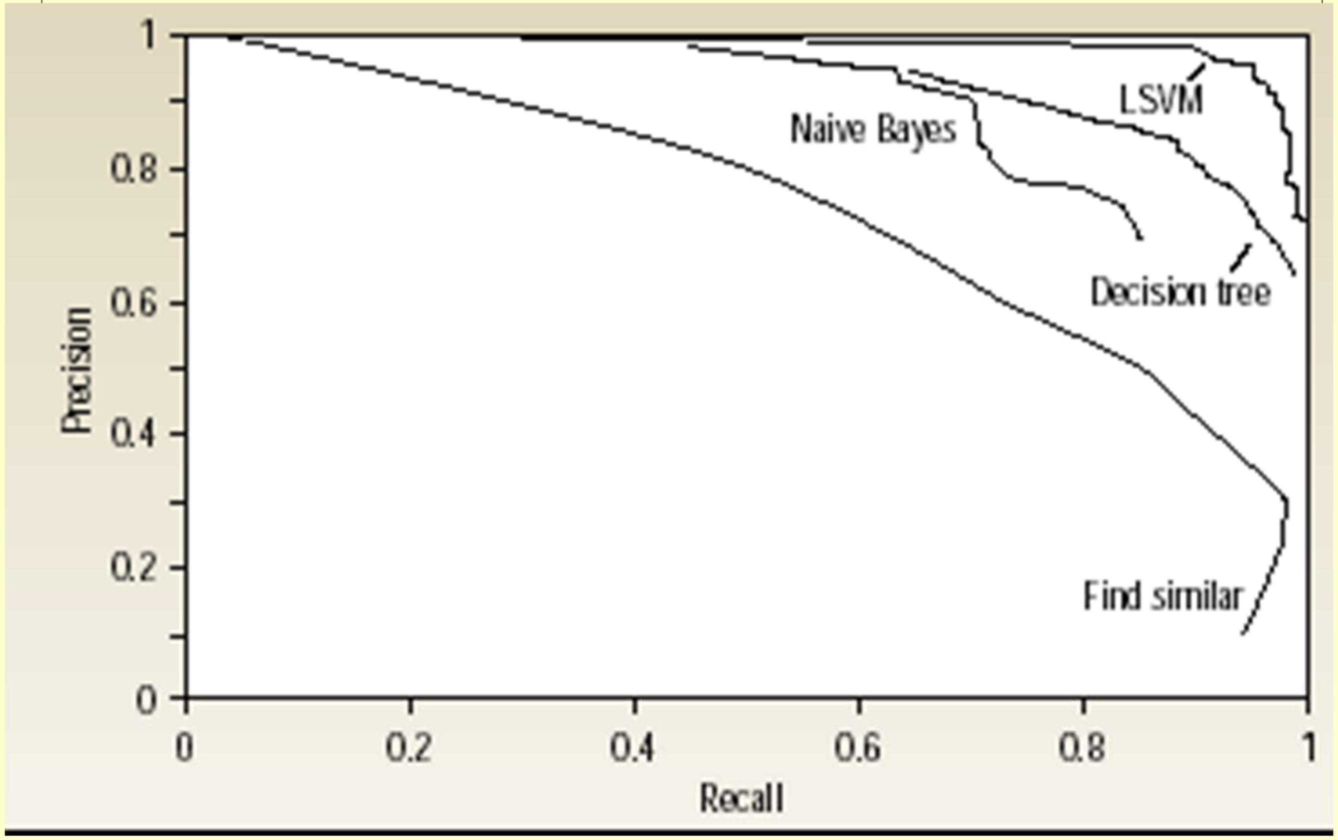
confidence ("interest" category) = 0.3*interest + 0.4*rate + 0.7*quarterly

- First comes from a PET; second from and SVM
- Text representation: BOW: mapping docs to large vectors indicating which word occurs in a doc; as many dimensions as words in the corpus (many more than in a given doc);
- often extended to frequencies (normalized) of *stemmed* words

Text classification

- Still a large number of features, so a stop list is applied, and some form of feature selection (e.g. based on info gain, or tf/idf) is done, down to 300 features
- Then a simple, linear SVM is used (experiments with poly. and RDF kernels indicated they are not much better than linear). One against all scheme is used
- What is a poly (e.g. level 2) kernel representing in text classification?
- Performance measured with micro-averaged break even point (explain)
- SVM obtained the best results, with DT second (on 10 cat.) and Bayes third. Other authors report better IB performance (findSim) than here

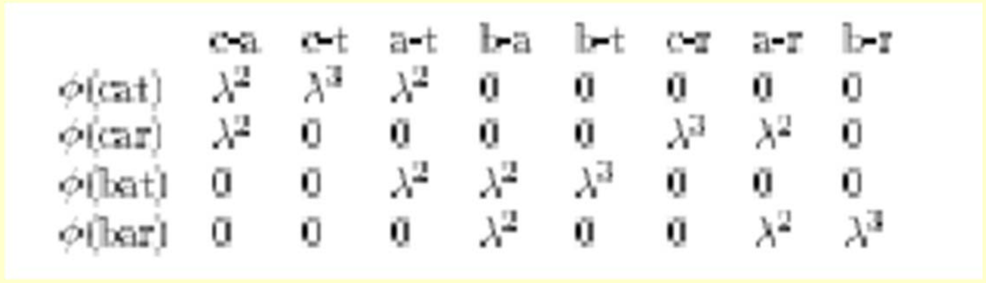
A ROC for the above experiments (class = “grain”)
ROC obtained by varying the threshold
threshold is learned
from values of $x \bullet w$ and discriminates between classes



How about another representation?

- N-grams = sequences of N consecutive characters, eg 3-grams is ‘support vector’ = sup, upp, ppo, por, …, tor
- Language-independent, but a large number of features (>>|words|)
- The more substrings in common between 2 docs, the more similar the 2 docs are
- What if we make these substring non-contiguous? With weight measuring non-contiguity? <u>**car**</u> – <u>**c**</u>ust<u>**ar**</u>d
- We will make ea substring a feature, with value depending on how frequently and how compactly a substring occurs in the text

- The latter is represented by a *decay factor* λ
- Example: cat, car, bat, bar



	c-a	c-t	a-t	b-a	b-t	c-r	a-r	b-r
ϕ(cat)	λ^2	λ^3	λ^2	0	0	0	0	0
ϕ(car)	λ^2	0	0	0	0	λ^3	λ^2	0
ϕ(bat)	0	0	λ^2	λ^2	λ^3	0	0	0
ϕ(bar)	0	0	0	λ^2	0	0	λ^2	λ^3

- Unnormalized K(car,cat)= λ^4, K(car,car)=K(cat,cat)=2 λ^4 + λ^6,normalized K(car,cat)= $\lambda^4/(2\lambda^4+ \lambda^6)= 1/(2+ \lambda^2)$
- Impractical (too many) for larger substrings and docs, but the kernel using such features can be calculated efficiently ('substring kernel' SSK) – maps strings (a whole doc) to a feature vector indexed by all *k*-tuples

- Value of the feature = sum over the occurrences of the k-tuple of a decay factor of the length of the occurrence
- Def of SSK: Σ is an alphabet; string = finite sequence of elems of Σ . $|s|$ = length of s; s[i:j] = substring of s. u is a subsequence of s if there exist indices $\mathbf{i}=(i_1,\dots,i_{|u|})$ with $1 \le i_1 < \dots < i_{|u|} \le |s|$ such that $u_j = s_{i_j}$ for $j=1,\dots,|u|$ ($u=s[\mathbf{i}]$ for short).
- Length $l(\mathbf{i})$ of of the subsequence *in s* is $i_{|u|} - i_1 + 1$ (span in s)
- Feature space mapping ϕ for s is defined by

$$\phi_u(s) = \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})}$$

for each $u \in \Sigma^n$ (set of all finite strings of length n): features measure the number of occurrences of subsequences in s weighed by their length ($\lambda \le 1$)

- The kernel can be evaluated in O(n|s||t|) time (see Lodhi paper)

Experimental results with SSK

- The method is NOT fast, so a subset of Reuters (*n*=470/380) was used, and only 4 classes: corn, crude, earn, acquisition
- Compared to the BOW representation (see earlier in these notes) with stop words removed, features weighed by tf/idf=log(1+*tf*)*log(*n/df*)
- F1 was used for the evaluation, C set experimentally
- Best *k* is between 4 and 7
- Performance comparable to a classifier based on k-grams (contiguous), and also BOW
- λ controls the penalty for gaps in substrings: best precision for high $\lambda = 0.7$. This seems to result in high similarity score for docs that share the same but *semantically different* words - WHY?
- Results on full Reuters not as good as with BOW, k-grams; the conjecture is that the kernel performs something similar to stemming, which is less important on large datasets where there is enough data to learn the 'sameness' of different inflections

Intro. to bioinformatics

- Bioinformatics = collection, archiving, organization and interpretation of biological data
- integrated *in vitro, in vivo, in silico*
- Requires understanding of basic genetics
- Based on
  - genomics,
  - proteomics,
  - transriptomics

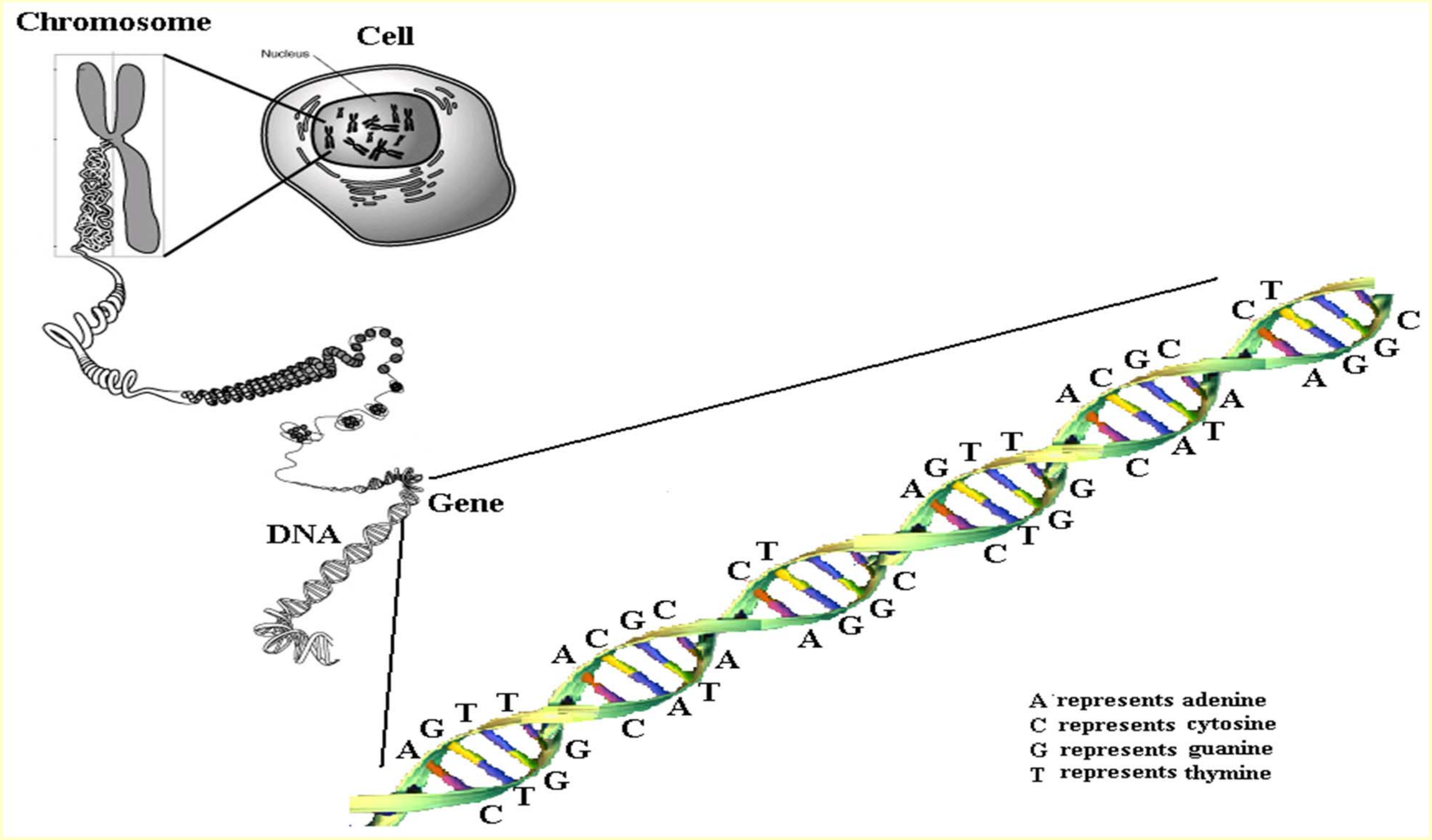
What is Bioinformatics?

- Bioinformatics is about integrating biological themes together with the help of computer tools and biological database.
- It is a “New” field of Science where mathematics, computer science and biology combine together to study and interpret genomic and proteomic information

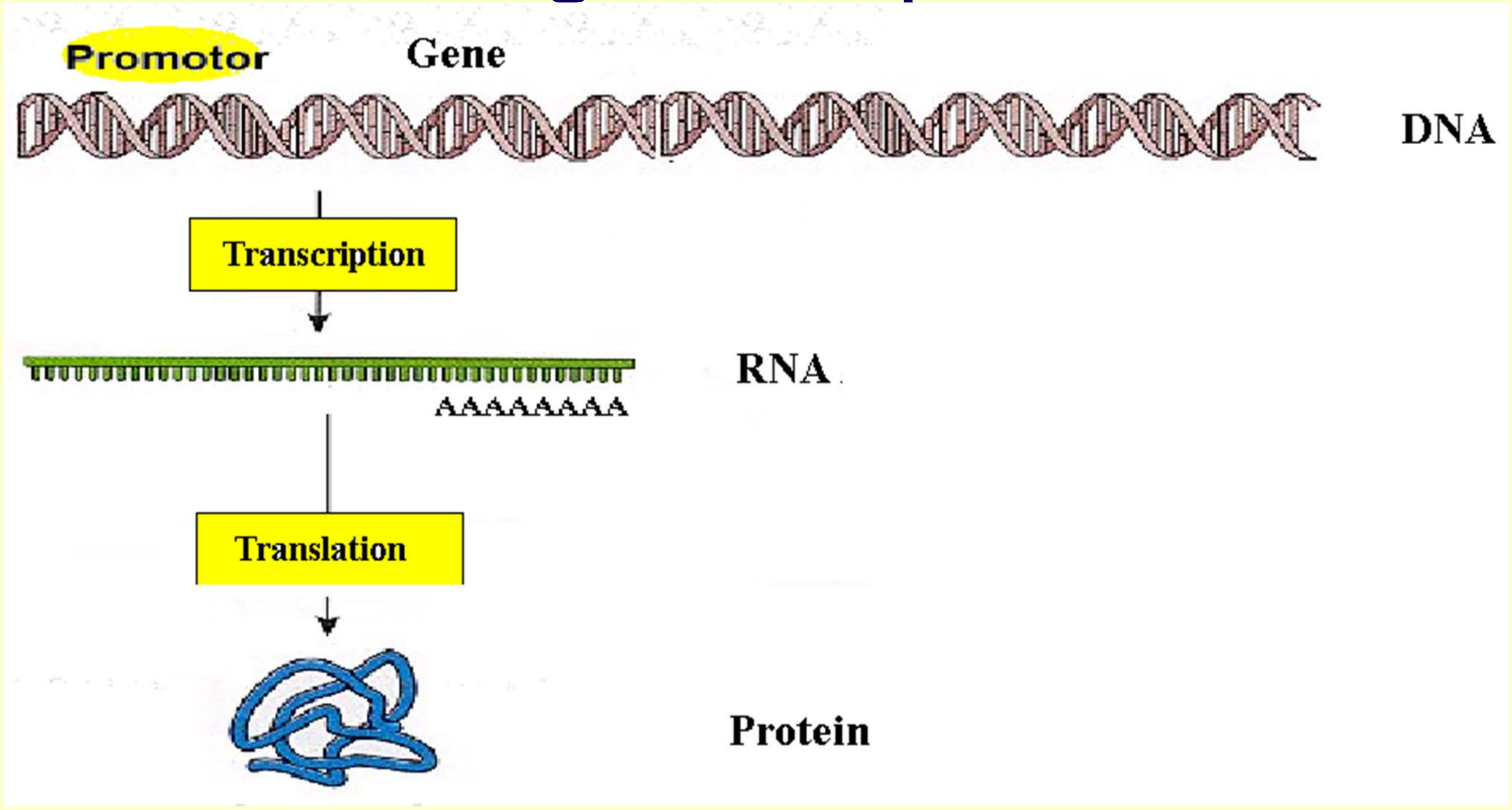
Basic biology

- Information in biology: DNA
- Genotype (hereditary make-up of an organism) and phenotype (physical/behavioral characteristics) (late 19th century)
- Biochemical structure of DNA – double helix – 1953; nucleotides A, C, G, T
- Progress in biology and IT made it possible to map the entire genomes: total genetic material of a species written with DNA code
- For a human, $3*10^9$ long
- Same in all the cells of a person

What is a gene?

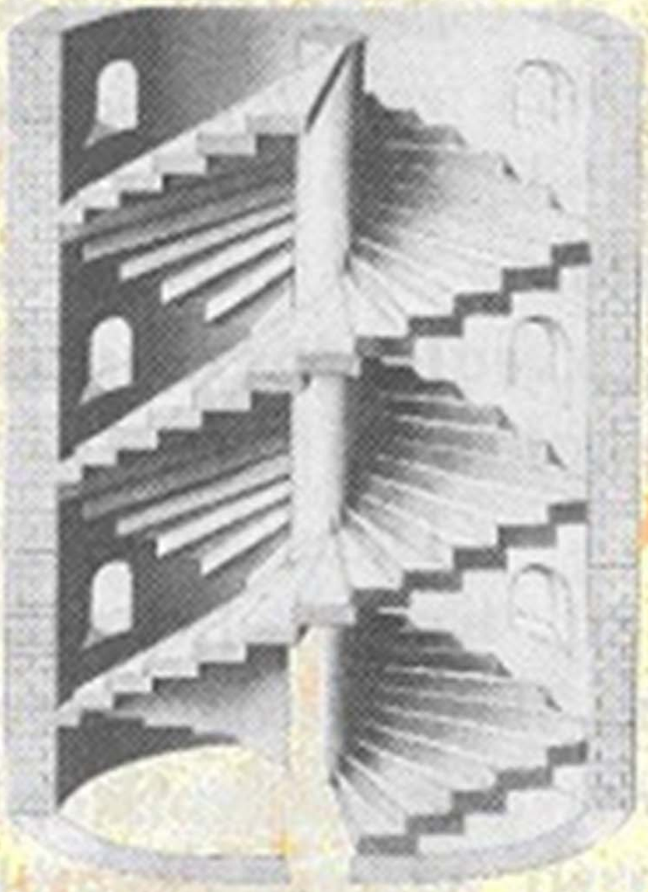


What is gene expression?



Genome

- The whole sequence in the alphabet A,C,G,T
- Human genome: > 3B DNA pairs, only 23K are encoding for proteins (only twice as much as a fly!)



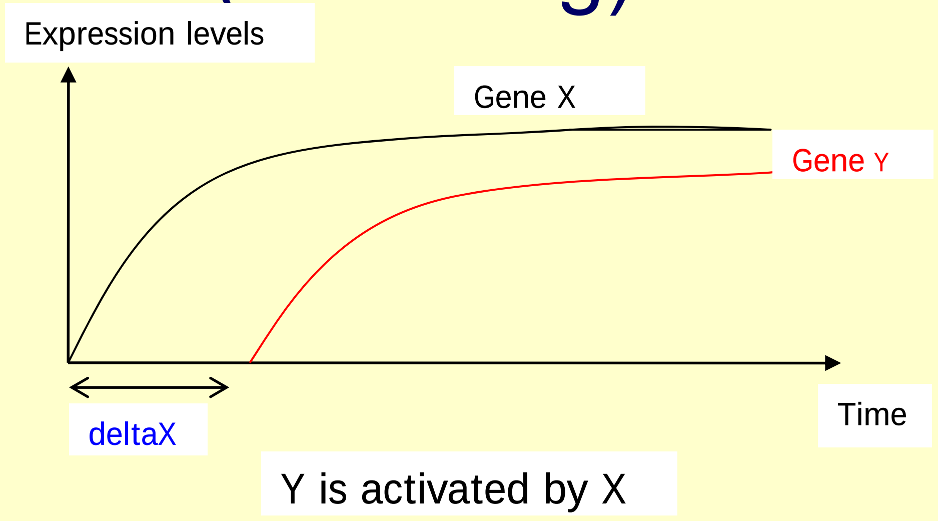
Chambord :
l'escalier à double volée

- Interesting to see if there are genes (functional elements of the genome) responsible for some aspects of the phenotype (e.g. an illness)
  - Testing
  - Cure
- Genes result in proteins:
- Gene $\xrightarrow{\text{RNA (transcription)}}$ protein

- We say that genes code for proteins
- In simple organisms (*prokaryotes*), high percentage of the genome are genes (85%)
- Is *eukaryotes* this drops: yeast 70%, fruit fly 25%, flowers 3%
- Databases with gene information: GeneBank/DDBL, EMBL
- Databases with Protein information:

  SwissProt, GenPept, TREMBL, PIR…

- Natural interest to find repetitive and/or common subsequences in genome: BLAST
- For this, it is interesting to study genetic expression (clustering):



- Activation + and Inhibition –

Bioinformatics application

- Coding sequences in DNA encode proteins.
- DNA alphabet: A, C, G, T. Codon = triplet of adjacent nucleotides, codes for one aminoacid.
- Task: identify where in the genome the coding (CDS, a coding sequence) starts (Translation Initiation Sites). Potential start codon is ATG.
- Classification task: does a sequence window around the ATG indicate a TIS?
- Each nucleotide is encoded by 5 bits, exactly one is set to 1, indicating whether the nucleotide is A, C, G, T, or unknown. So the dimension *n* of the input space = 1000 for window size 100 to left *and* right of the ATG sequence.
- Positive and negative windows are provided as the training set
- This representation is typical for the kind of problem where SVMs do well

- What is a good feature space for this problem? how about including in the kernel some prior – domain – knowledge? Eg:
- Dependencies between distant positions are not important or are known not to exist
- Compare, at each sequence position, two sequences locally in a window of size 2*l*+1 around that position, with decreasing weight away from the centre of the window:

$$\mathrm{win}_p(\mathrm{x},\mathrm{y}) = \Big(\sum_{j=-l}^{+l} p_j \mathrm{match}_{p+j}(\mathrm{x},\mathrm{y})\Big)^{d_1}$$

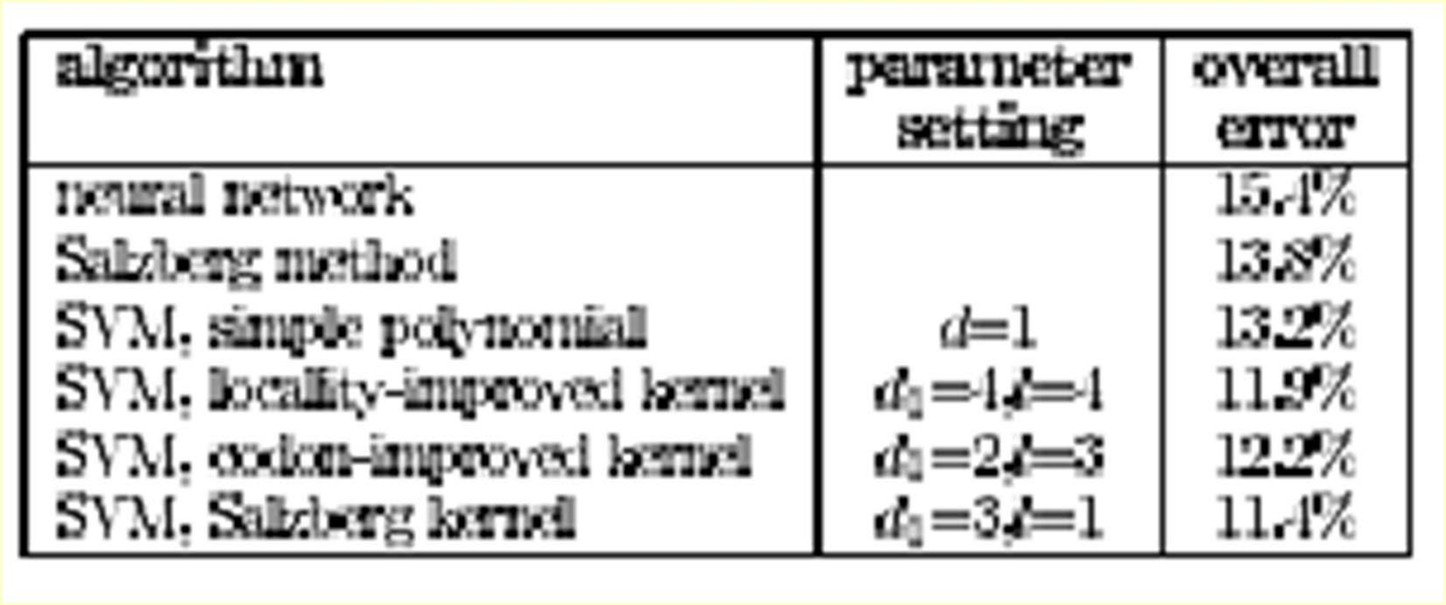
- Where d_1 is the order of importance of local (within the window) correlations, and match_{p+j} is 1 for matching nucleotides at position *p*+*j*, 0 otherwise

- Window scores are summed over the length of the sequence, and correlations between up to *d_2* windows are taken into account:

$$\mathrm{k}(\mathrm{x},\mathrm{y}) = \left(\sum_{p=1}^{l} \mathrm{win}_p(\mathrm{x},\mathrm{y})\right)^{d_2}$$

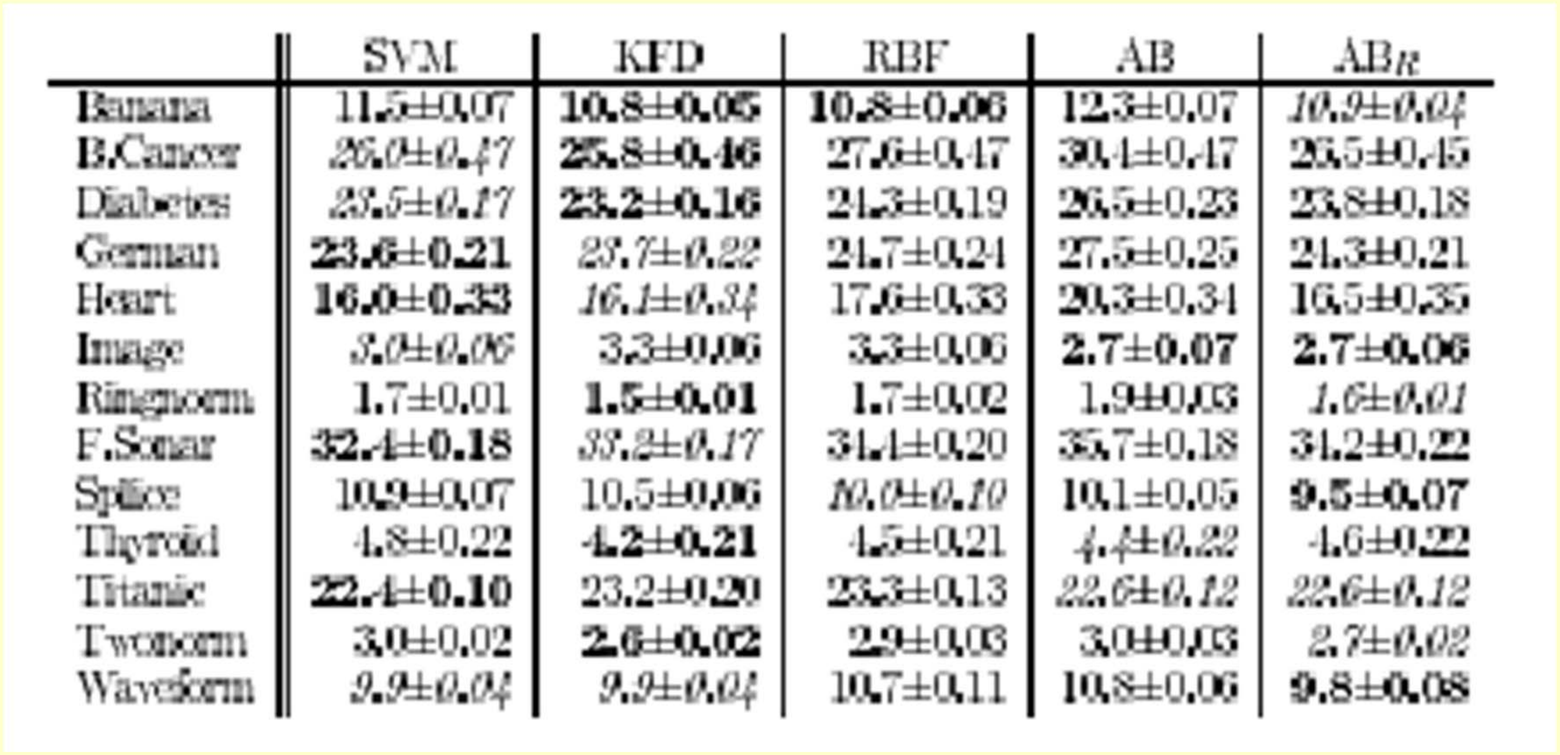
- Also it is known that the codon below the TIS is a CDS: CDS shifted by 3 nucleotides is still a CDS
- Trained with 8000 patterns and tested with 3000

Results



algorithm	parameter setting	overall error
neural network		15.4%
Salzberg method		13.8%
SVM, simple polynomial	d=1	13.2%
SVM, locality-improved kernel	d_1=4,d=4	11.9%
SVM, codon-improved kernel	d_1=2,d=3	12.2%
SVM, Salzberg kernel	d_1=3,d=1	11.4%

Further results on UCI benchmarks



	SVM	KFD	RBF	AB	AB_R
Banana	11.5±0.07	**10.8±0.05**	**10.8±0.06**	12.3±0.07	*10.9±0.04*
B.Cancer	*26.0±0.47*	**25.8±0.46**	27.6±0.47	30.4±0.47	26.5±0.45
Diabetes	*23.5±0.17*	**23.2±0.16**	24.3±0.19	26.5±0.23	23.8±0.18
German	**23.6±0.21**	*23.7±0.22*	24.7±0.24	27.5±0.25	24.3±0.21
Heart	**16.0±0.33**	*16.1±0.34*	17.6±0.33	20.3±0.34	16.5±0.35
Image	*3.0±0.06*	3.3±0.06	3.3±0.06	**2.7±0.07**	**2.7±0.06**
Ringnorm	1.7±0.01	**1.5±0.01**	1.7±0.02	1.9±0.03	*1.6±0.01*
F.Sonar	**32.4±0.18**	*33.2±0.17*	34.4±0.20	35.7±0.18	34.2±0.22
Splice	10.9±0.07	10.5±0.06	*10.0±0.10*	10.1±0.05	**9.5±0.07**
Thyroid	4.8±0.22	**4.2±0.21**	4.5±0.21	*4.4±0.22*	4.6±0.22
Titanic	**22.4±0.10**	23.2±0.20	23.3±0.13	*22.6±0.12*	*22.6±0.12*
Twonorm	3.0±0.02	**2.6±0.02**	2.9±0.03	3.0±0.03	*2.7±0.02*
Waveform	*9.9±0.04*	*9.9±0.04*	10.7±0.11	10.8±0.06	**9.8±0.08**

- View
- `http://videolectures.net/acai05_taylor_svkm/`