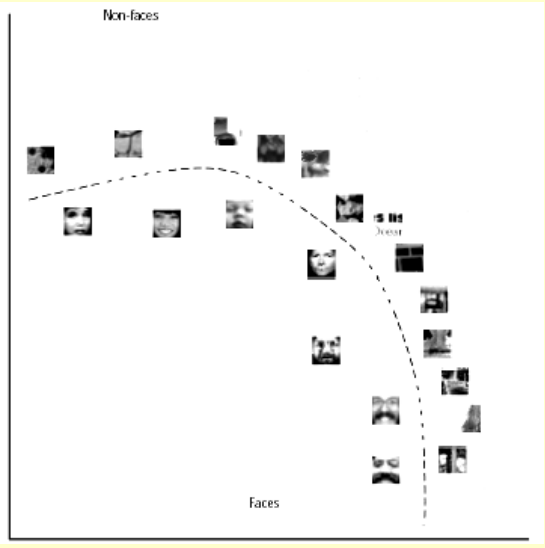
SVM cont'd

- 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)

Applications – face detection [IEEE INTELLIGENT SYSTEMS]

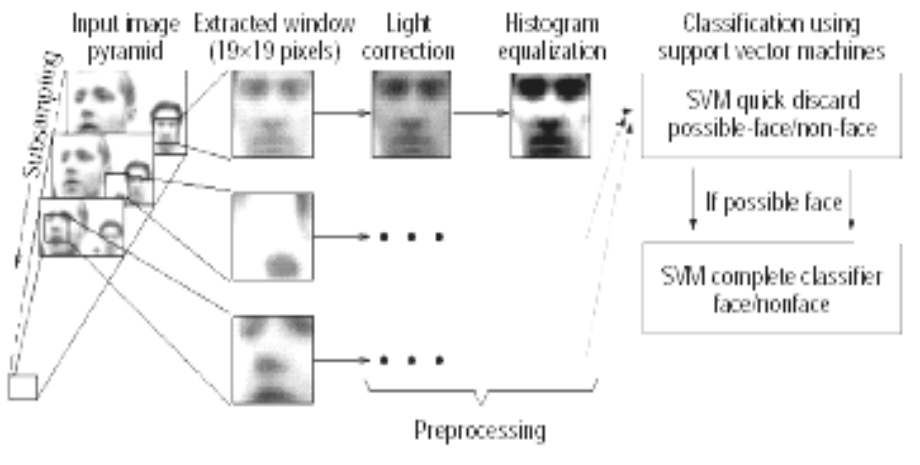
- The task: to find a rectangle containing a face in an image applicable in face recognition, surveillance, HCI etc. Also in medical image processing and structural defects
- Difficult task – variations that are hard toi represent explicitly (hair, moustache, glasses)
- Cast as a classification problem: 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

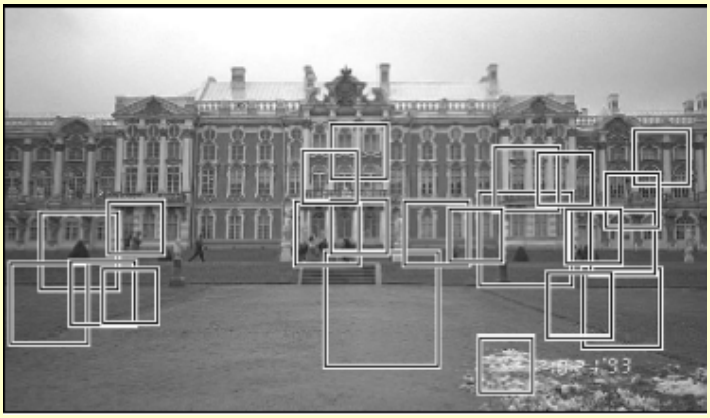


- SVM performing face detection –support vectors are faces and non-faces
- Examples are 19x19 pixels, class +1 or -1
- SVM: 2nd degree polynomial with slack variables
- Representation tricks: masking out near-boundary area - 361->283, removes noise
- illumination correction: reduction of light and shadow
- Discretization of pixel brightness by histogram equalization

Face detection – system architecture



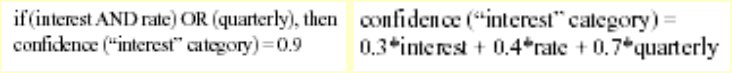
Bootstrapping : using the system on images with no faces and storing false positives to use as negative examples in later training



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.

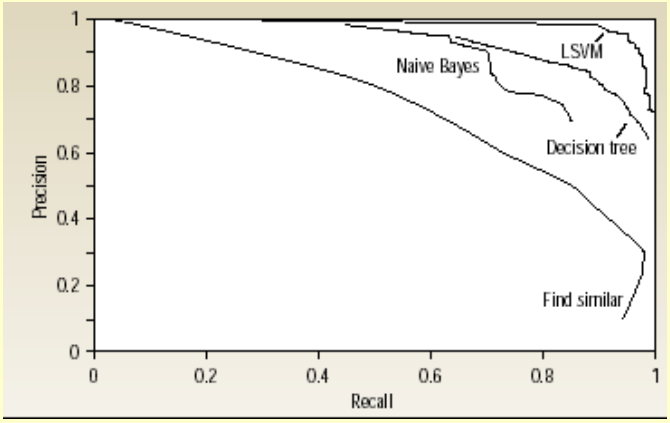


- First comes from a PET; second from and SVM
- Text representation: BOW: mapping docs to large vectors indicating which word occurs in a doc; as many dimensions as words in the corpus (many more than in a given doc);
- often extended to frequencies (normalized) of *stemmed* words

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? **car** – **c**ust**ar**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)= λ^4/(2λ^4+ λ^6)= 1/(2+ λ^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 $\mathbf{i}=(i_1,\ldots,i_{|u|})$ with $1\le i_1<\ldots< i_{|u|} \le |s|$ such that $u_j = s_{i_j}$ for j=1,…,|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 λ = 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 onlarge datasets where there is enough data to learn the 'samness' of different inflections

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 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 negaite 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:

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

- 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:

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

- Also it is known that the codon below the TIS is a CDS: CDS shifted by 3 nucleotides is still a TDS
- 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	*25.0±0.47*	**25.8±0.46**	27.6±0.47	30.4±0.47	25.5±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**