Advanced Machine Learning & Perception

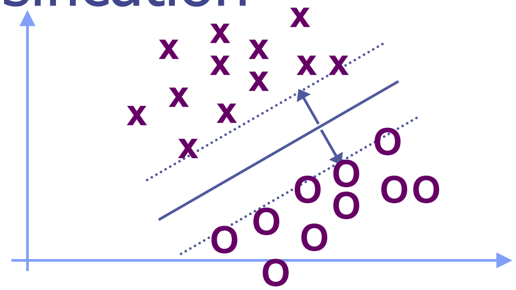
Instructor: Tony Jebara

SVM Feature & Kernel Selection

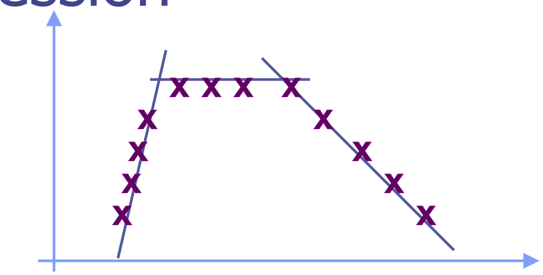
- SVM Extensions
- Feature Selection (Filtering and Wrapping)
- SVM Feature Selection
- SVM Kernel Selection

SVM Extensions

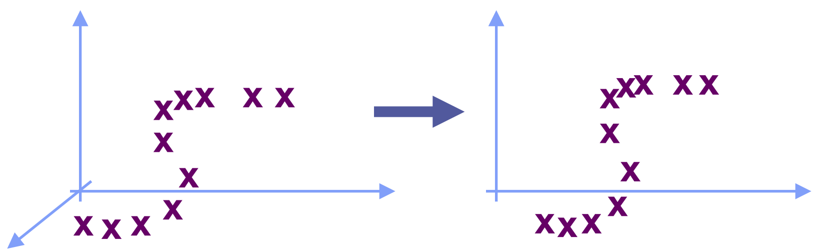
Classification



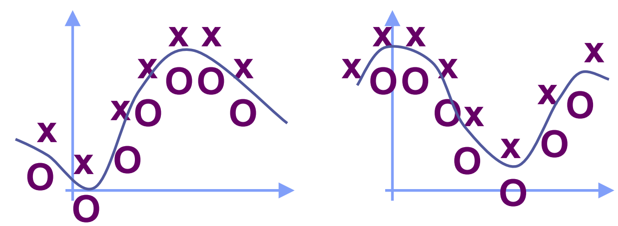
Regression



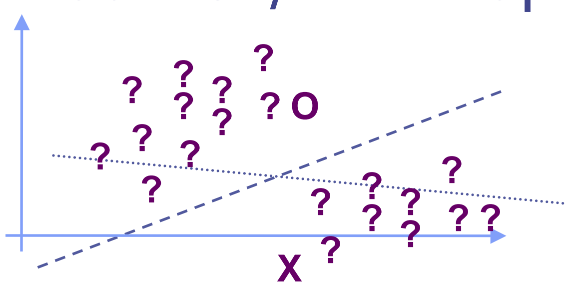
Feature/Kernel Selection



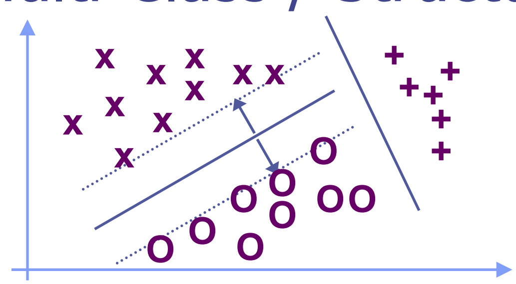
Meta/Multi-Task Learning



Transduction/Semi-supervised

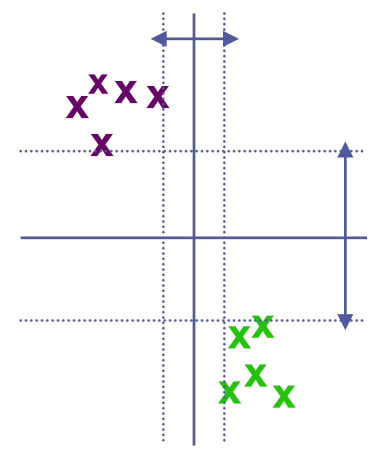


Multi-Class / Structured



Feature Selection & Sparsity

- Isolates interesting dimensions of data for a given task
- Reduces complexity of data
- Augments Sparse Vectors (SVMs) with Sparse Dimensions
- Can also *Improve Generalization*
- Example: find subset of d features from D dims that give largest margin SVM?



$$L\left(\vec{x} \mid \theta\right) = \sum_{i=1}^{D} s_i \vec{x}_i \theta_i + b \qquad s_i \in \left\{0,1\right\} \;\&\; \sum_{i=1}^{D} s_i = d$$

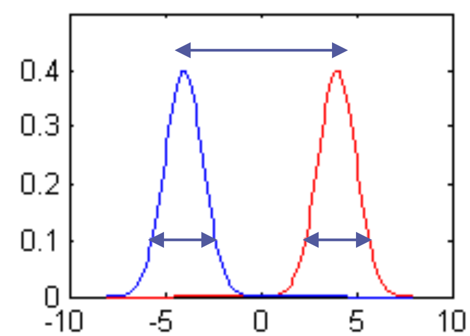
- Typically needs exponential search: 1000 choose 10 if we consider all possible subsets of dimensions
- How to do this efficiently (and jointly) with SVM estimation? Two classical approaches: *Filtering* & *Wrapping*

Feature Selection: Filtering

- Filtering: find/eliminate some features before even training your classifier (before induction) as a pre-processing.
- Wrapping: find/eliminate some features by evaluating their accuracy after you train your classifier (after induction).
- Fisher Information Criterion: Compute score below for each feature i=1…D. Keep the top d features

$$Fisher(i) = \left| \frac{\mu_i^+ - \mu_i^-}{(\sigma_i^+)^2 + (\sigma_i^-)^2} \right|$$

$$\mu_i^+ = \frac{1}{T_+} \sum_{t \in +} \vec{x}_i^t \qquad (\sigma_i^+)^2 = \frac{1}{T_+} \sum_{t \in +} (\vec{x}_i^t - \mu_i^+)^2$$



Like putting a Gaussian on each class in each 1 dimension to compute their spread. The Gaussian assumption may be wrong! Only measures how linearly separable data is.

Feature Selection: Filtering

- Pearson Correlation Coefficients: score how similar or redundant two features are. Can then remove redunancies or remove features that are too correlated on average.

$$Pearson(i,j) = \left| \frac{\sum_t \left(\vec{x}_i^t - \mu_i\right)\left(\vec{x}_j^t - \mu_j\right)}{\left(T+1\right)\sigma_i \sigma_j} \right|$$

...again Gaussian only

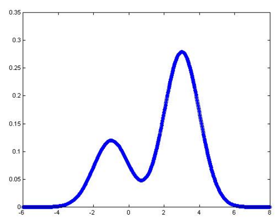
- Kolmogorov-Smirnov Test: non-parametric, more general than Gaussian but only 1 feature at a time.

For each feature, compute the cumulative density function over both classes then over the single class. Find KS score as follows, keep top d features.

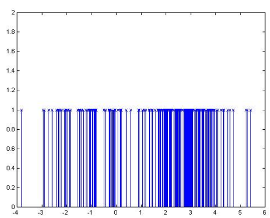
$$KolmogorovSmirnov(i) = \sqrt{T} \sup_q \left(\hat{P}\left\{\vec{x}_i \leq q\right\} - \hat{P}\left\{\vec{x}_i \leq q \mid y = 1\right\}\right)$$

Feature Selection: Filtering

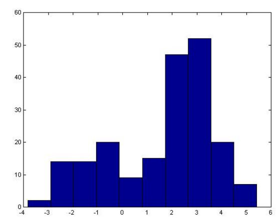
- Kolmogorov-Smirnov example:



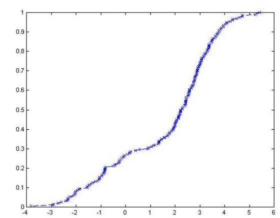
$$P\left(\vec{x}_i\right)$$



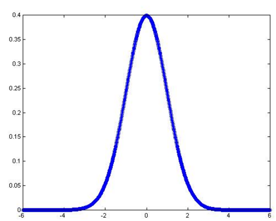
$$\left\{\vec{x}_i^1,\ldots\right\}$$



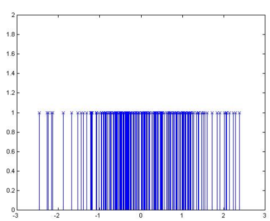
$$\hat{P}\left(\vec{x}_i\right)$$



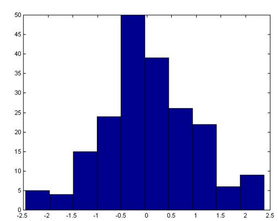
$$\hat{P}\left(\vec{x}_i \le q\right)$$



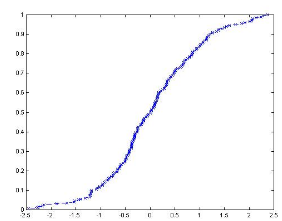
$$P\left(\vec{x}_i \mid y=1\right)$$



$$\left\{\vec{x}_i^1,\ldots \mid y=1\right\}$$

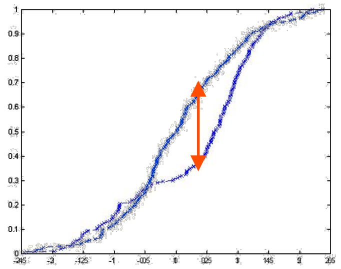


$$\hat{P}\left(\vec{x}_i \mid y=1\right)$$



$$\hat{P}\left(\vec{x}_i \le q \mid y=1\right)$$

$$KS\left(i\right) = \sqrt{T}\,\sup_q \left(\hat{P}\left\{\vec{x}_i \le q\right\} - \hat{P}\left\{\vec{x}_i \le q \mid y=1\right\}\right)$$



Feature Selection: Wrapping

- Wrapping: use accuracy of resulting classifier to drive the feature selection $f(\vec{x}) = w^T \phi(\vec{x} \bullet \vec{s}) + b$
- Dot is elementwise product of x with binary vector s
- Note: more features usually improves training accuracy.
- So, pre-specify the maximum number (or %) of features
- Or, optimize generalization bound (SRM vs. ERM)
- Margin & Radius Bound (like VC-bound):

$$E\left\{P_{err}\right\} \leq \frac{1}{T} E\left\{\frac{R^2}{M^2}\right\} = \frac{1}{T} E\left\{R^2 W^2(\alpha)\right\}$$

Expectations over datasets

- Better Span Bound: (if SV's don't change when doing leave-one out cross-validation, i.e. removing point p)

$$E\left\{P_{err}^{T-1}\right\} \leq \frac{1}{T} E\left\{\sum_{p=1}^{T} u\left(\frac{\alpha_p}{\left(K_{SV}^{-1}\right)_{pp}} - 1\right)\right\}$$

u() is step function Ksv is Gram matrix of only support vectors

SVM Feature Selection

- Margin & Radius Bound: optimize via gradient descent
- Assume selection vector s is given: $k\left(\vec{x}_t,\vec{x}_{t'}\right) = k\left(\vec{x}_t \bullet s, \vec{x}_{t'} \bullet s\right)$
- Compute R^2 and betas via:

$$R^2 = \max_\beta \sum_t \beta_t k\left(\vec{x}_t,\vec{x}_t\right) - \sum_{t,t'} \beta_t \beta_{t'} k\left(\vec{x}_t,\vec{x}_{t'}\right) \qquad s.t. \sum_t \beta_t = 1 \;\; \beta_t \geq 0$$

- Compute W^TW and alphas via:

$$\max_\alpha \sum_t \alpha_t - \sum_{t,t'} \alpha_t \alpha_{t'} y_t y_{t'} k\left(\vec{x}_t,\vec{x}_{t'}\right) \; s.t. \alpha_t \in \left[0,C\right], \sum_t \alpha_t y_t = 0$$

- Assume switches are continuous, take derivatives of R^2/M^2:

$$\frac{\partial R^2 W^2}{\partial s_i} = R^2 \frac{\partial W^2}{\partial s_i} + W^2 \frac{\partial R^2}{\partial s_i}$$

$$\frac{\partial R^2}{\partial s_i} = \sum_t \beta_t \frac{\partial k\left(\vec{x}_t,\vec{x}_t\right)}{\partial s_i} - \sum_{t,t'} \beta_t \beta_{t'} \frac{\partial k\left(\vec{x}_t,\vec{x}_{t'}\right)}{\partial s_i}$$

$$\frac{\partial W^2}{\partial s_i} = -\sum_{t,t'} y_t y_{t'} \alpha_t \alpha_{t'} \frac{\partial k\left(\vec{x}_t,\vec{x}_{t'}\right)}{\partial s_i}$$

SVM Feature Selection

- Use chain rule to get gradient of kernel with respect to s.
- E.g. RBF kernel

$$
\begin{aligned}
\frac{\partial k\left(\vec{x}_t, \vec{x}_{t'}\right)}{\partial s_i} &= \frac{\partial}{\partial s_i}\left(\exp\left(-\tfrac{1}{2}\left\|\vec{x}_t .* s - \vec{x}_{t'} .* s\right\|^2\right)\right) \\
&= \frac{\partial}{\partial s_i}\left(\exp\left(-\tfrac{1}{2}\sum_{j=1}^{D} s_j^2\left(\vec{x}_t\left(j\right) - \vec{x}_{t'}\left(j\right)\right)^2\right)\right) \\
&= \exp\left(-\tfrac{1}{2}\sum_{\substack{j=1 \\ j\neq i}}^{D} s_j^2\left(\vec{x}_t\left(j\right) - \vec{x}_{t'}\left(j\right)\right)^2\right)\frac{\partial}{\partial s_i}\left(\exp\left(-\tfrac{1}{2} s_i^2\left(\vec{x}_t\left(i\right) - \vec{x}_{t'}\left(i\right)\right)^2\right)\right) \\
&= \exp\left(-\tfrac{1}{2}\sum_{\substack{j=1 \\ j\neq i}}^{D} s_j^2\left(\vec{x}_t\left(j\right) - \vec{x}_{t'}\left(j\right)\right)^2\right) \\
&\quad \times \exp\left(-\tfrac{1}{2} s_i^2\left(\vec{x}_t\left(i\right) - \vec{x}_{t'}\left(i\right)\right)^2\right)\left(-s_i\left(\vec{x}_t\left(i\right) - \vec{x}_{t'}\left(i\right)\right)^2\right)
\end{aligned}
$$

SVM Feature Selection

- Assemble all calculations to get gradient vector over s

$$\frac{\partial R^2}{\partial s_i} = \sum_t \beta_t \frac{\partial k\left(\vec{x}_t, \vec{x}_t\right)}{\partial s_i} - \sum_{t,t'} \beta_t \beta_{t'} \frac{\partial k\left(\vec{x}_t, \vec{x}_{t'}\right)}{\partial s_i}$$

$$\frac{\partial W^2}{\partial s_i} = -\sum_{t,t'} y_t y_{t'} \alpha_t \alpha_{t'} \frac{\partial k\left(\vec{x}_t, \vec{x}_{t'}\right)}{\partial s_i}$$

- Given the old s value, $s = \left[0\ 1\ 1\ 0\right]^T$ the gradient is:

$$\frac{\partial R^2 W^2}{\partial s_i} = R^2 \frac{\partial W^2}{\partial s_i} + W^2 \frac{\partial R^2}{\partial s_i} = 92.4 \begin{bmatrix} 0.4 \\ 0.2 \\ -3.2 \\ 2.4 \end{bmatrix} + 25.4 \begin{bmatrix} -0.3 \\ 3.1 \\ 3.5 \\ -2.3 \end{bmatrix}$$

- Take a small step to drive down the term (against gradient)

SVM Feature Selection

- Synthesized from mixture of Gaussian data
- Feature selection improves classifier & speeds it up

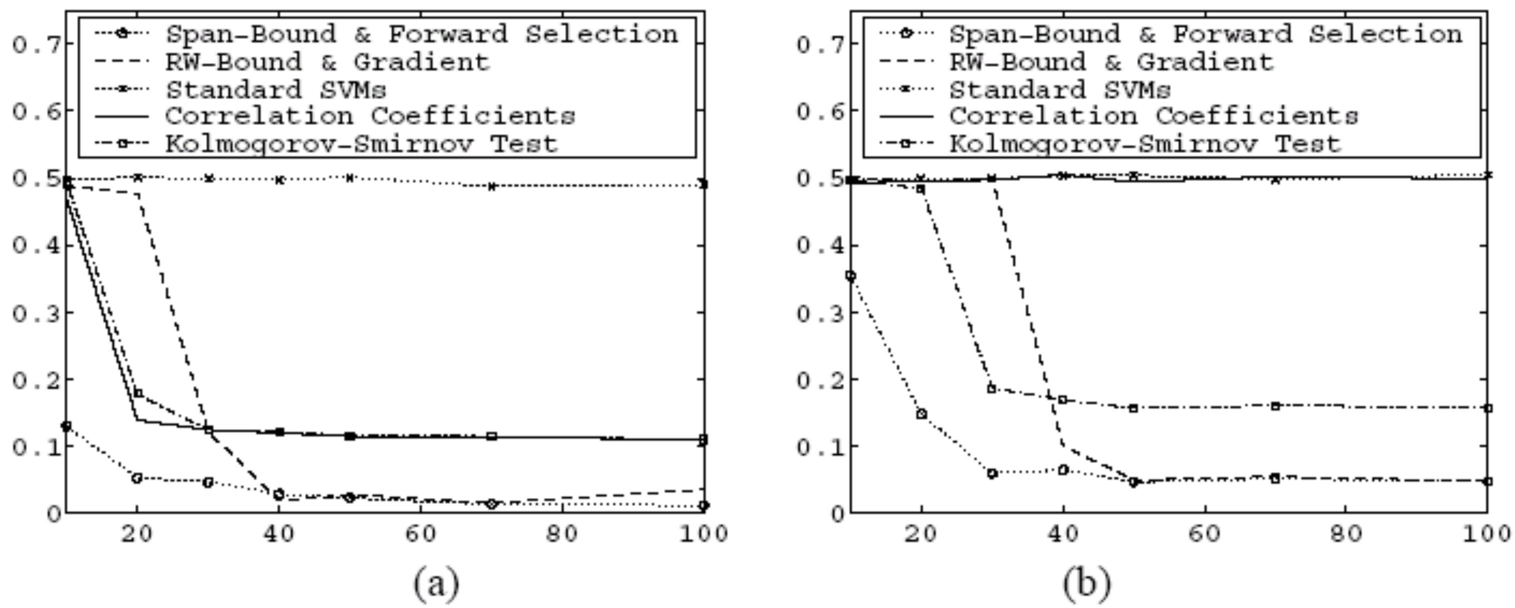
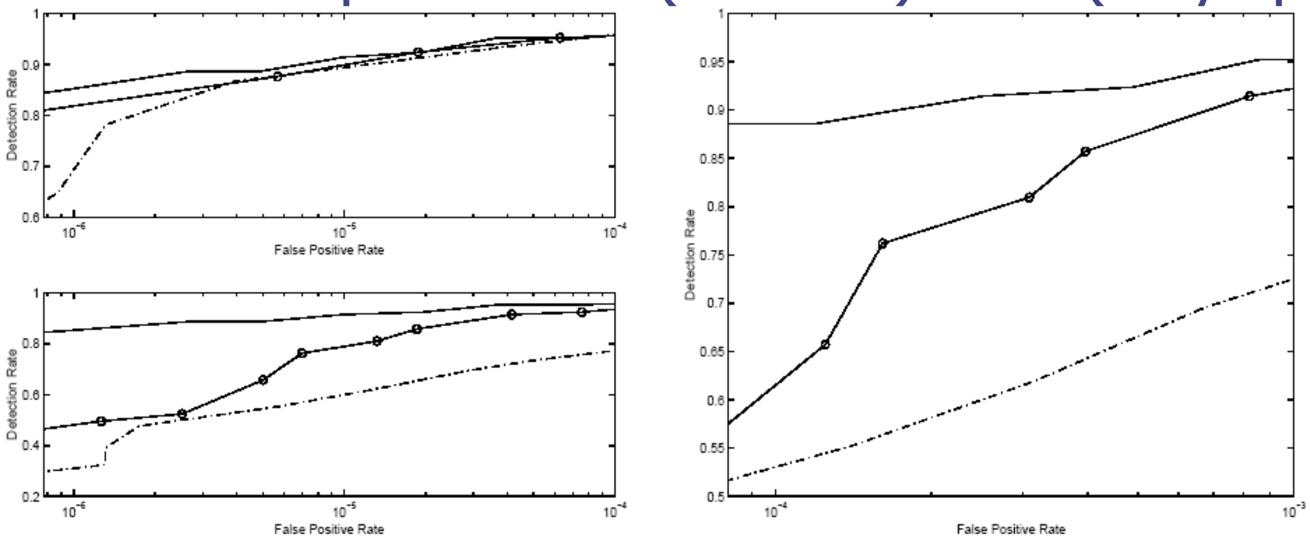


Figure 1: A comparison of feature selection methods on (a) a linear problem and (b) a nonlinear problem both with many irrelevant features. The x-axis is the number of training points, and the y-axis the test error as a fraction of test points.

SVM Feature Selection

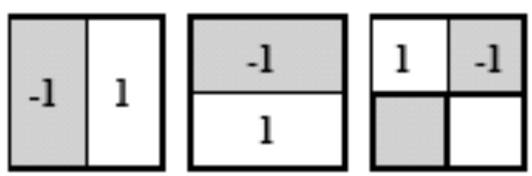
- Real face & pedestrian (wavelet) data (only speedup)

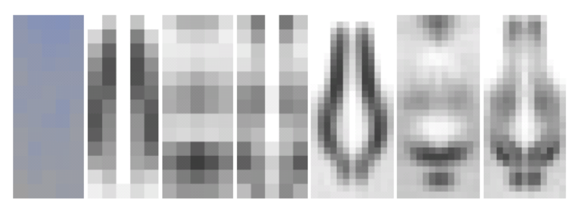


(a) (b)

Figure 2: The solid line is using all features, the solid line with a circle is our feature selection method (minimizing R^2W^2 by gradient descent) and the dotted line is the Fisher score. (a)The top ROC curves are for 725 features and the bottom one for 120 features for face detection. (b) ROC curves using all features and 120 features for pedestrian detection.

- Wavelet basis:





SVM Kernel Selection

- We are given d=1…D base kernels to use in an SVM

$$k_1(\vec{x},\vec{x}'),k_2(\vec{x},\vec{x}'),\ldots,k_D(\vec{x},\vec{x}')$$

- How do we pick the best ones or a combination of them?

$$k_{FINAL}(\vec{x},\vec{x}') = k_4(\vec{x},\vec{x}') + k_9(\vec{x},\vec{x}') + k_{12}(\vec{x},\vec{x}')$$

- It we only had to use 1 kernel, try D different SVMs…
- To pick 5 out of 10 kernels, need 10 choose 5 = 252 SVMs!
- Even worse is picking a weighted combination of kernels where the alpha weights are positive

$$k_{FINAL}(\vec{x},\vec{x}') = \sum_{i=1}^{D} \alpha_i k_i(\vec{x},\vec{x}')$$

- Define the alignment between two kernel matrices as

$$A(K_1,K_2) = \frac{\langle K_1,K_2\rangle}{\sqrt{\langle K_1,K_1\rangle}\sqrt{\langle K_2,K_2\rangle}} \quad where \langle K_1,K_2\rangle = \sum_{i,j=1}^{N} k_1(\vec{x}_i,\vec{x}_j)k_2(\vec{x}_i,\vec{x}_j)$$

SVM Kernel Selection

- We want a kernel matrix K that aligns with the labels matrix

$$\max_K A\left(K, yy^T\right)$$

- This can be written equivalently as the solution below:

$$\max_K \left\langle K, yy^T \right\rangle \; s.t. \; \left\langle K, K \right\rangle = 1, K \succeq 0$$

- This can all be written as a semidefinite program (SDP)

$$\max_K \left\langle K, yy^T \right\rangle \; s.t. \begin{pmatrix} A & K^T & 0 & 0 \\ K & I & 0 & 0 \\ 0 & 0 & 1 - tr\left(A\right) & 0 \\ 0 & 0 & 0 & K \end{pmatrix} \succeq 0$$

- Unfortunately, this can give a trivial solution…

SVM Kernel Selection

- Instead, force K to be a conic combination of base kernels:

$$\max_K \left\langle K, yy^T \right\rangle$$

$$s.t. \begin{pmatrix} A & K^T & 0 & 0 \\ K & I & 0 & 0 \\ 0 & 0 & 1 - tr\left(A\right) & 0 \\ 0 & 0 & 0 & K \end{pmatrix} \succeq 0$$

$$PLUS\ldots \quad K = \sum_{i=1}^{D} \alpha_i K_i$$

- This is simpler than an SDP, just a second order cone program (faster code)

Table 1. Margin and number of test-set errors (TSE) for SVMs trained and tested with the initial kernel matrices K_1, K_2, K_3 and with the optimal kernel matrix K^*, learned using semi-definite programming (12) with $c = \sum_i \text{trace}(K_i)$. A dash means that no hard margin classifier could be found.

	K_1	K_2	K_3	K^*
Breast cancer	$d = 2$	$\sigma = 0.5$		
margin	0.010	0.136	-	0.300
TSE	19.7	28.8		11.4
Sonar	$d = 2$	$\sigma = 0.1$		
margin	0.035	0.198	0.006	0.352
TSE	15.5	19.4	21.9	13.8
Heart	$d = 2$	$\sigma = 0.5$		
margin	-	0.159	-	0.285
TSE		49.2		36.6

Feature vs. Kernel Selection

- Linear feature selection can be done via kernel selection!

$$f(\vec{x}) = w^T(\vec{x} \bullet \vec{s}) + b \quad \text{via} \quad K = \sum_{i=1}^{D} s_i K_i$$

... where only a few s values are 1 and most are zero

- Define the base kernels

$$k_1(\vec{x}, \vec{x}'), k_2(\vec{x}, \vec{x}'), \ldots, k_D(\vec{x}, \vec{x}')$$

to be: $k_i(\vec{x}, \vec{x}') = \vec{x}(i)\vec{x}'(i)$

- For example, in a linear SVM the classifier is:

$$f(\vec{x}) = \sum_t \alpha_t y_t k_{FINAL}(\vec{x}, \vec{x}_t) + b = \sum_t \alpha_t y_t \sum_i s_i k_i(\vec{x}, \vec{x}_t) + b$$

$$= \sum_t \alpha_t y_t \sum_i s_i \vec{x}(i)\vec{x}_t(i) + b = w^T(\vec{x} \bullet \vec{s}) + b$$