Advanced Machine Learning & Perception

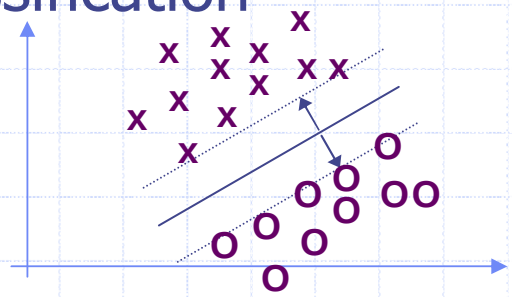
Instructor: Tony Jebara

Topic 9

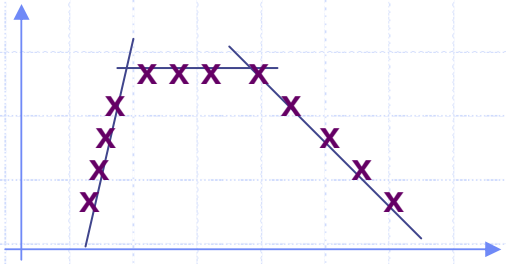
- SVM Extensions
- Feature Selection (Filtering and Wrapping)
- SVM Feature Selection
- SVM Kernel Selection
- MED Feature Selection
- MED Kernel Selection
- Meta-Learning and Multi-Task Learning

SVM Extensions

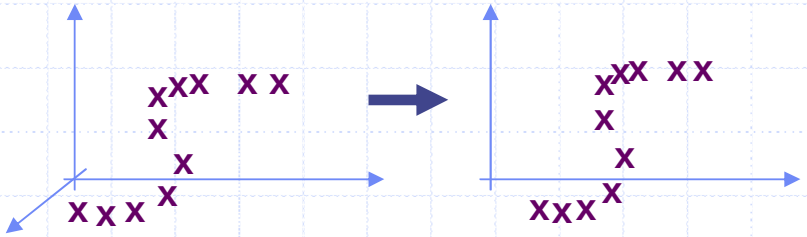
Classification



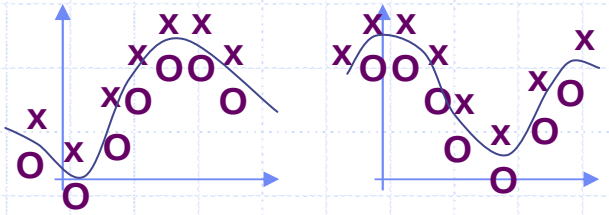
Regression



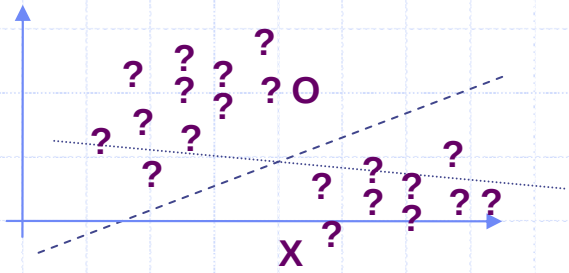
Feature/Kernel Selection



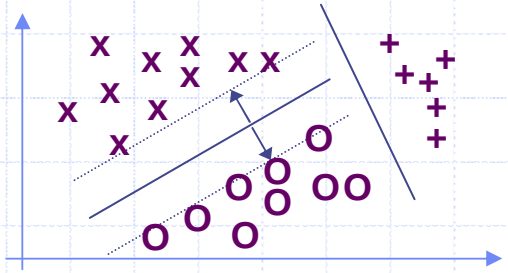
Meta/Multi-Task Learning



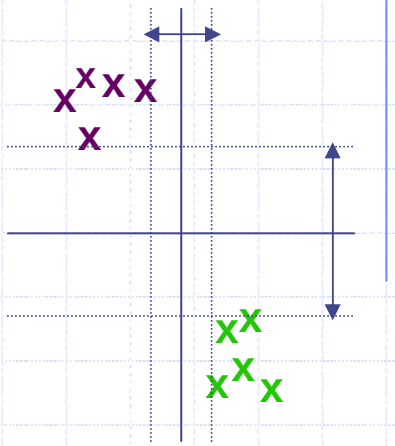
Transduction



Multi-Class / Structured



Feature Selection



- Isolate interesting dimensions of data for given task
- Reduces complexity of data
- Augments Sparse Vectors (SVMs) with Sparse Dimensions
- Also *Improves Generalization*
- Example: find subset of N features from D dims that give largest margin SVM?

$$L(X \mid \theta) = \sum_{i=1}^{D} s_i X_i \theta_i + b \qquad s_i \in \{0,1\} \;\&\; \sum_{i=1}^{D} s_i = N$$

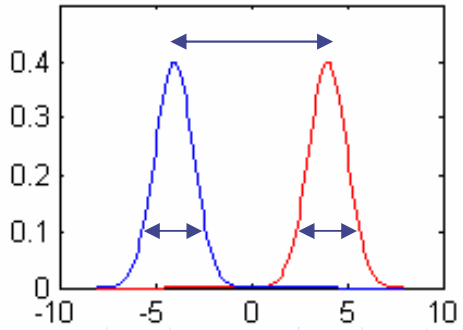
- Typically needs exponential search: 1000 choose 10 if we consider all possible subsets of dimensions
- How to do this efficiently (and possible jointly) with SVM parameter estimation? *Filtering* or *Wrapping (more work)*...

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 N features

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

$$\mu_i^+ = \frac{1}{T_+} \sum_{t \in +} X_i^t \qquad \left(\sigma_i^+\right)^2 = \frac{1}{T_+} \sum_{t \in +} \left(X_i^t - \mu_i^+\right)^2$$



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.

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

...again Gaussian only

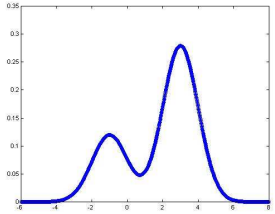
- Kolmogorov-Smirnov Test: non-parametric, more general than Gaussian but only 1 feature at a time.

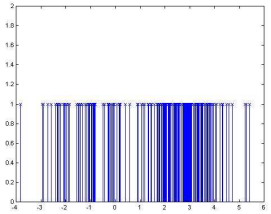
  For each feature, compute the cumulative density function over both classes then over the single class. Find KS score as follows, keep top N features.

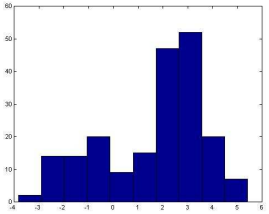
$$KolmogorovSmirnov(i) = \sqrt{T}\,\sup_q \left(\hat{P}\left\{X_i \le q\right\} - \hat{P}\left\{X_i \le q \mid y = 1\right\} \right)$$

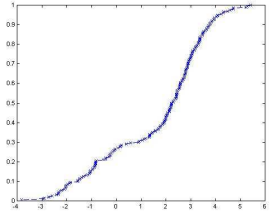
Feature Selection: Filtering

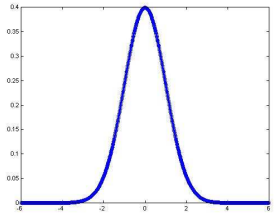
- Kolmogorov-Smirnov example:

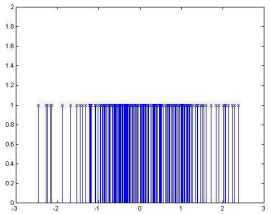

$P\left(X_i\right)$

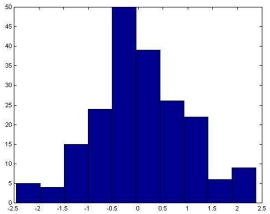

$\left\{X_i^1,\ldots\right\}$

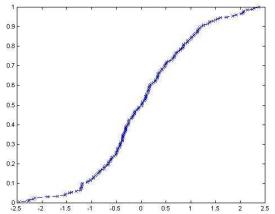

$\hat{P}\left(X_i\right)$


$\hat{P}\left(X_i \le q\right)$

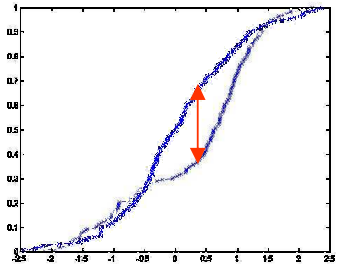

$P\left(X_i \mid y=1\right)$


$\left\{X_i^1,\ldots \mid y=1\right\}$


$\hat{P}\left(X_i \mid y=1\right)$


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

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



Feature Selection: Wrapping

- Wrapping: use accuracy of resulting classifier to drive the feature selection $f(X) = w^T \phi(X .* \vec{s}) + b$
- But (obviously) using more features almost always helps improve training accuracy.
- Either pre-specify the maximum number (or %) of features
- Or, optimize generalization bound (SRM vs. ERM)
- Margin & Radius Bound (like VC-bound):

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

Expectations over datasets

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

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

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: $K\left(X_t, X_{t'}\right) = K\left(X_t .* \vec{s}, X_{t'} .* \vec{s}\right)$
- Compute R^2 and betas via:

$$R^2 = \max_\beta \sum_t \beta_t K\left(X_t, X_t\right) - \sum_{t,t'} \beta_t \beta_{t'} K\left(X_t, X_{t'}\right) \;\; s.t. \sum_t \beta_t = 1 \;\; \beta_t \geq 0$$

- Compute W^TW and alphas via:

$$\max_\alpha \sum_t \alpha_t - \sum_{t,t'} \alpha_t \alpha_{t'} y_t y_{t'} K\left(X_t, X_{t'}\right) \; s.t. \alpha_t \in \left[0, C\right], \sum_t \alpha_t y_t = 0$$

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

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

$$\frac{\partial R^2}{\partial s_i} = \sum_t \beta_t \frac{\partial K\left(X_t, X_t\right)}{\partial s_i} - \sum_{t,t'} \beta_t \beta_{t'} \frac{\partial K\left(X_t, X_{t'}\right)}{\partial s_i}$$

$$\frac{\partial W^2}{\partial s_i} = -\sum_{t,t'} y_t y_{t'} \alpha_t \alpha_{t'} \frac{\partial K\left(X_t, X_{t'}\right)}{\partial s_i}$$

SVM Feature Selection

- Use chain rule to get gradient of kernel with respect to s.
- Example: RBF kernel…

$$\frac{\partial K\left(X_t, X_{t'}\right)}{\partial s_i} = \frac{\partial}{\partial s_i}\left(\exp\left(-\tfrac{1}{2}\left\|X_t .* s - X_{t'} .* s\right\|^2\right)\right)$$

$$= \frac{\partial}{\partial s_i}\left(\exp\left(-\tfrac{1}{2}\sum_{j=1}^{D} s_j^2\left(X_t(j) - X_{t'}(j)\right)^2\right)\right)$$

$$= \exp\left(-\tfrac{1}{2}\sum_{\substack{j=1\\ j\neq i}}^{D} s_j^2\left(X_t(j) - X_{t'}(j)\right)^2\right)$$

$$\times \frac{\partial}{\partial s_i}\left(\exp\left(-\tfrac{1}{2} s_i^2\left(X_t(i) - X_{t'}(i)\right)^2\right)\right)$$

$$= \exp\left(-\tfrac{1}{2}\sum_{\substack{j=1\\ j\neq i}}^{D} s_j^2\left(X_t(j) - X_{t'}(j)\right)^2\right)$$

$$\times \exp\left(-\tfrac{1}{2} s_i^2\left(X_t(i) - X_{t'}(i)\right)^2\right)\left(-s_i\left(X_t(i) - X_{t'}(i)\right)^2\right)$$

SVM Feature Selection

- Assemble all calculations to get gradient vector over s

$$\frac{\partial R^2}{\partial s_i} = \sum_t \beta_t \frac{\partial K(X_t, X_t)}{\partial s_i} - \sum_{t,t'} \beta_t \beta_{t'} \frac{\partial K(X_t, X_{t'})}{\partial s_i}$$

$$\frac{\partial W^2}{\partial s_i} = -\sum_{t,t'} y_t y_{t'} \alpha_t \alpha_{t'} \frac{\partial K(X_t, X_{t'})}{\partial s_i}$$

- Given old s:

$$s = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

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

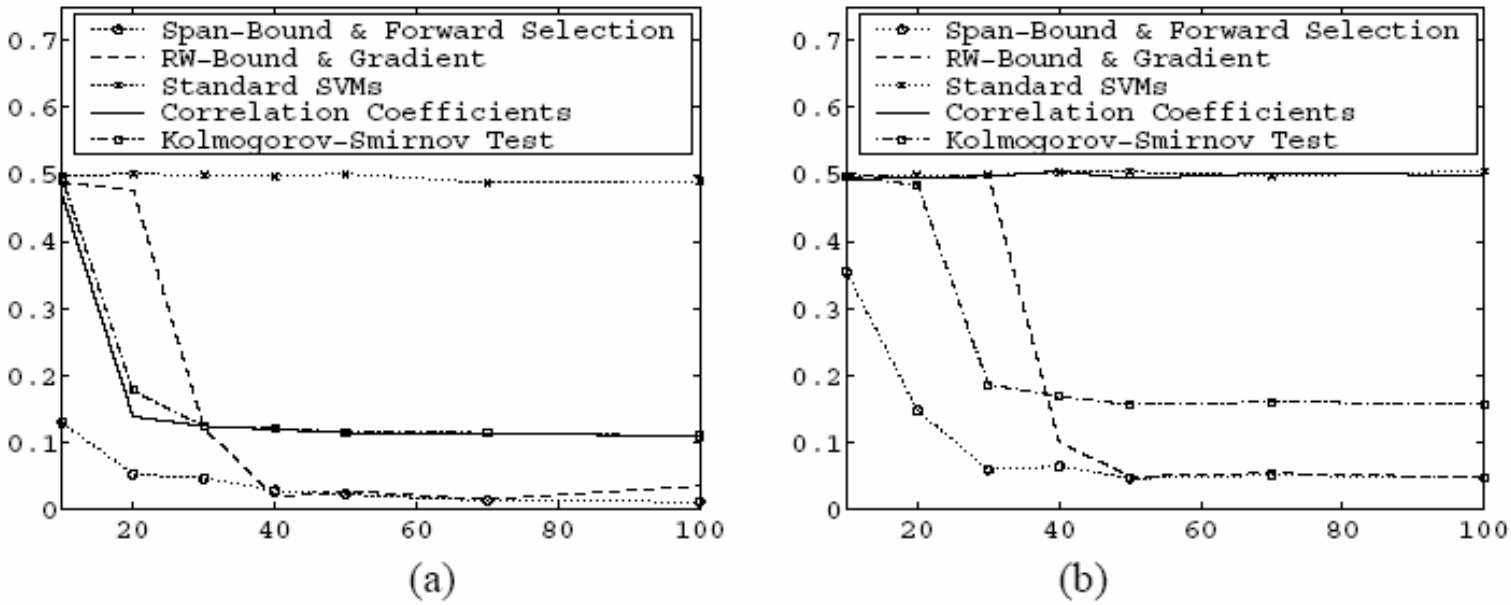
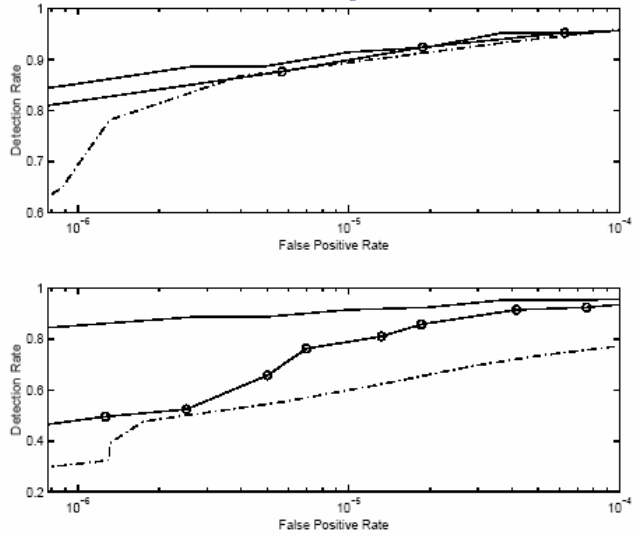


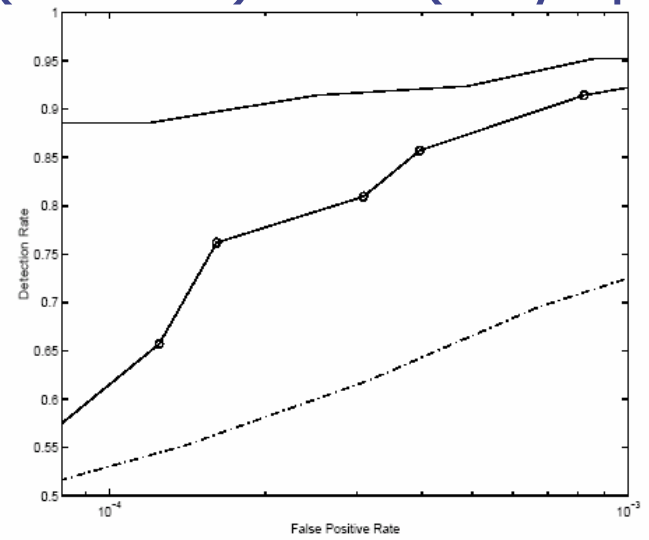
Figure 1: A comparison of feature selection methods on (a) a linear problem and (b) a nonlinear problem both with many irrelevant features. The x-axis is the number of training points, and the y-axis the test error as a fraction of test points.

SVM Feature Selection

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



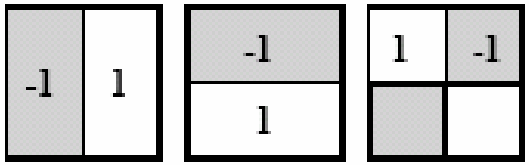
(a)

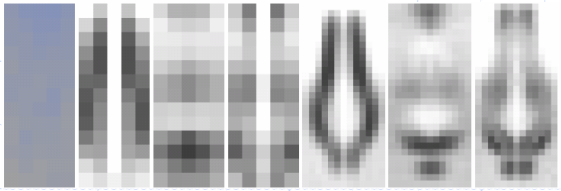


(b)

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 possible kernels to use in an SVM

$$k_1(X,X'), k_2(X,X'), \ldots, k_D(X,X')$$

- How do we pick the best ones?

$$k_{FINAL}(X,X') = k_4(X,X') + k_9(X,X') + k_{12}(X,X')$$

- 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

$$k_{FINAL}(X,X') = \sum_{d=1}^{D} \alpha_d k_d(X,X')$$

- It turns out this can be solved as a semidefinite program!
- Define the 'alignment' between two kernel matrices:

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

SVM Kernel Selection

- We want to find a kernel matrix K which is most aligned with the labels matrix:

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

but want matrix to be a convex combo: $K = \sum_{d=1}^{D} \alpha_d K_d$

or at least positive semi-definite: $K \succeq 0$

- This is equivalent to: $\max_K \left\langle K, yy^T \right\rangle \; s.t. \; \left\langle K, K \right\rangle = 1, K \succeq 0$

- To write this as an SDP:

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

SVM Kernel Selection

- Or we can explore conic combination of kernels:

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

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

$$AND... \quad K = \sum_{d=1}^{D} \alpha_d K_d$$

- 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

MED Support Vector Machine

- Recall MED technique for SVMs, etc.
- Many θ solutions valid, find $p(\theta)$
- Find $P(\Theta,\gamma)$ minimizing $KL\left(P\middle\|P_0\right)+\sum_t \xi_t$ subject to $\int P(\Theta)\left[y_t L\left(X_t;\Theta\right)-1\right]d\Theta \geq \xi_t,\ \forall t$
- Resulting classifier is: $\hat{y}=sign\int P(\Theta)\,L(X;\Theta)\,d\Theta$
- Solution:

$$P(\Theta)=\frac{1}{Z(\lambda)}P_0(\Theta)\exp\left(\sum_t \lambda_t\left[y_t L\left(X_t;\Theta\right)-1\right]\right)$$

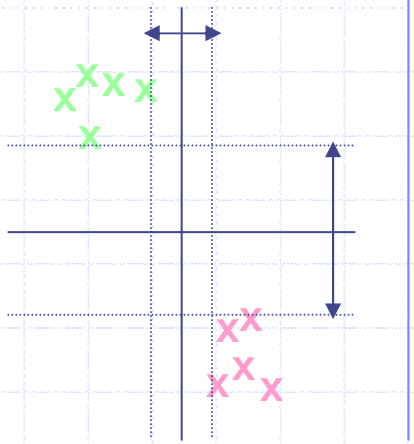
- Partition:

$$Z(\lambda)=\int_\Theta P_0(\Theta)\exp\left(\sum_t \lambda_t\left[y_t L\left(X_t;\Theta\right)-1\right]\right)d\Theta$$

- Objective: maximize $J(\lambda)=\sum_t \lambda_t-\frac{1}{2}\sum_{t,t'}\lambda_t\lambda_{t'}y_t y_{t'}\left(X_t^T X_{t'}\right)$

with SVM constraints: λ_t in [0,C] and $\Sigma_t \lambda_t y_t=0$

MED Feature Selection



- Goal: pick 100 of 10000 features to get largest margin classifier (NP)
- Turn features on/off via binary switches

$$s_i \in \{0,1\}$$

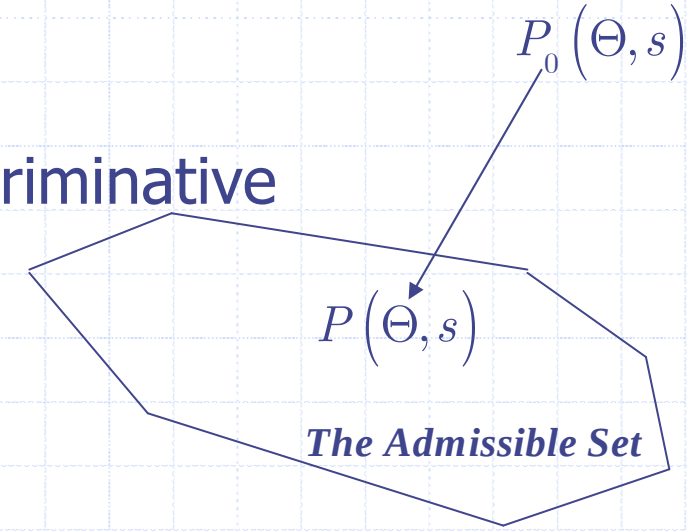
Switch Prior: Bernoulli distribution $P_{s,0}(s_i) = \rho^{s_i}(1-\rho)^{1-s_i}$

rho parameter varies pruning level

$$L(X;\Theta) = \sum_i s_i \theta_i X_i + b$$

MED uniquely & efficiently finds discriminative feature subset, analytic partition fn:

$$Z(\lambda) = \int P(\Theta, s)$$

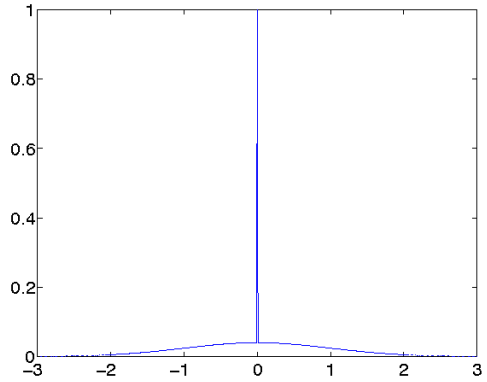


MED Feature Selection

- Modify parameters to include $L(X;\Theta) = \sum_{i=1}^{D} s_i \theta_i X_i + \theta_0$ a binary ON / OFF Switch
- The model $\Theta = \{\theta_0,\ldots,\theta_D, s_1,\ldots,s_D\}$ contains structural $s_i \in \{0,1\}$ parameters to aggressively prune features.

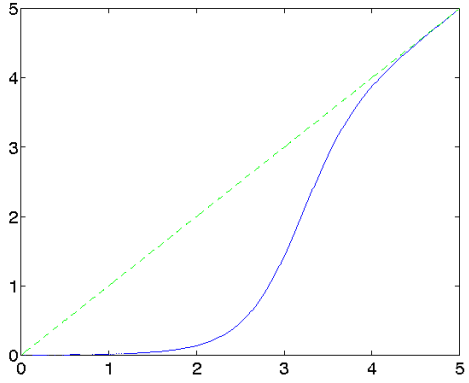
Prior: $P_0(\Theta) = P_{0,\theta_0}(\theta_0) P_{0,\theta}(\theta) P_{0,s}(s) = P_{0,\theta_0}(\theta_0) N(\theta \mid 0, I) \prod_i P_0(s_i)$

Switch Prior: Bernoulli distribution $P_{s,0}(s_i) = \rho^{s_i}(1-\rho)^{1-s_i}$ rho parameter selects no pruning to aggressive pruning



Prior on $s_i \theta_i$

Aggressive attenuation of linear coefficients at low values (rho=.01).

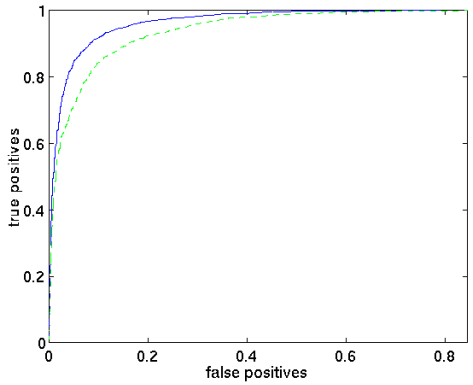


MED Feature Selection

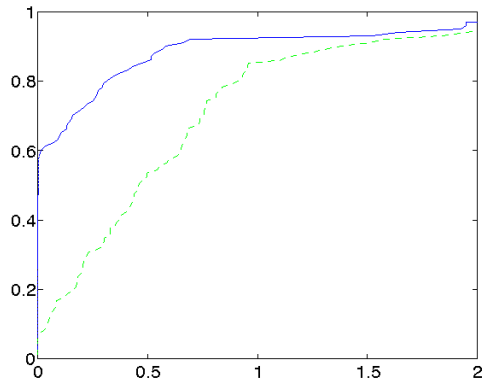
Objective function $J(\lambda) = \sum_t \lambda_t - \sum_{i=1}^{D} \log\left[1 - \rho + \rho e^{\frac{1}{2}\left(\sum_t \lambda_t y_t X_{t,i}\right)^2}\right]$

With SVM constraints: λ_t in [0,C] and $\Sigma_t \lambda_t y_t = 0$

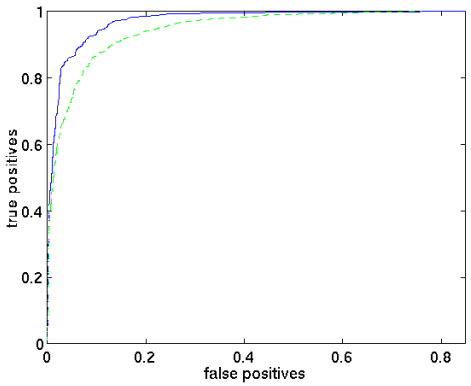
DNA Data: 2-class, 100 element binary vectors. Train/Test=500/4724



ROC of DNA Splice Site
100 Features
Original 25xGATC



CDF of Linear Coeffs
DNA Splice Site
100 Features



ROC DNA Splice Site
~5000 Features
Quadratic Kernel

Dashed line: ρ = 0.99999

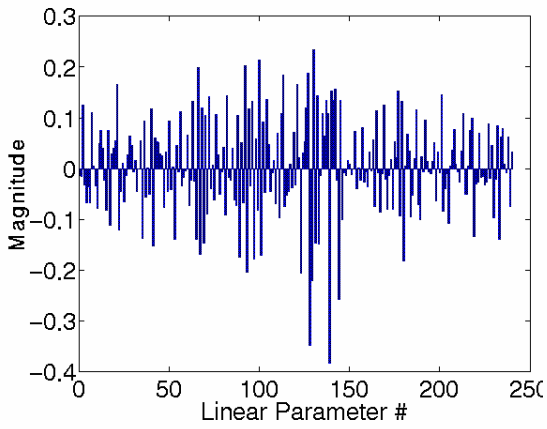
Solid line: ρ = 0.00001

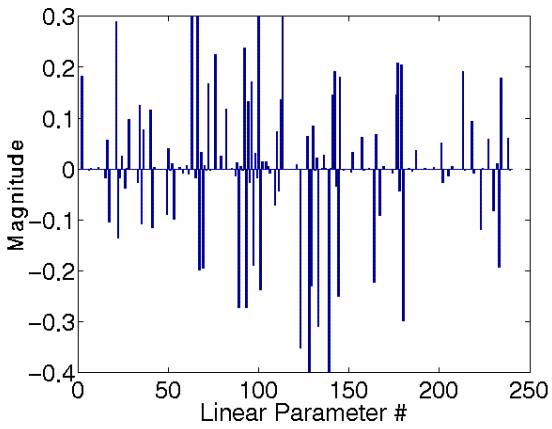
MED Feature Selection

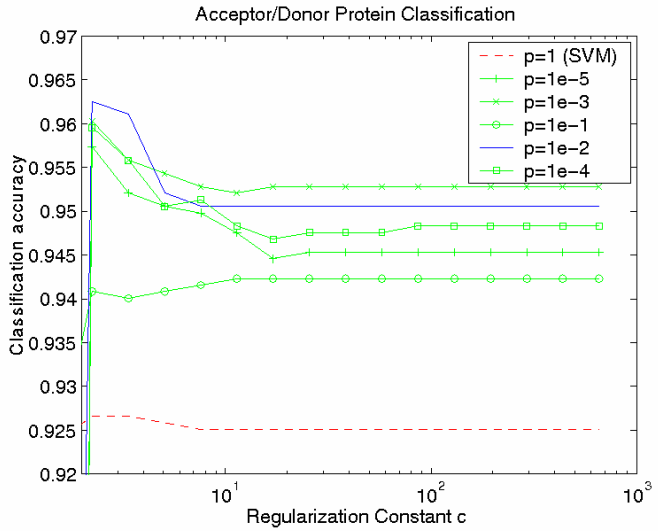
$$J(\lambda) = \sum_t \lambda_t - \sum_{i=1}^{D} \log\left[1 - \rho + \rho e^{\frac{1}{2}\left(\sum_t \lambda_t y_t X_{t,i}\right)^2}\right]$$

λ constrained to [0,c] hyper-cube with constraint $\sum_t \lambda_t y_t = 0$

Example: Intron-Exon Protein Classification:
UCI: 240 dims; 200 train, 1300 test







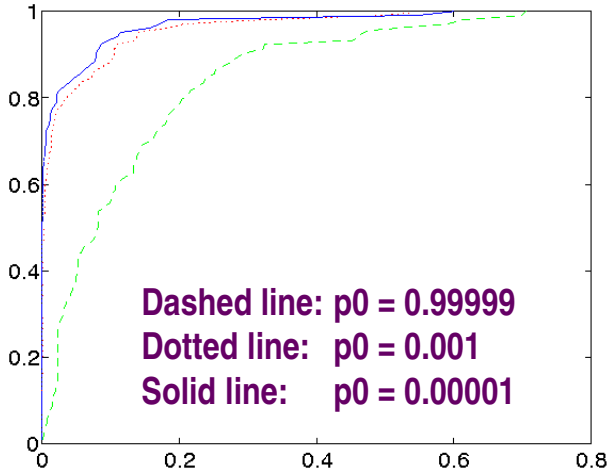
MED Feature Selection

- Introduce switches into regression and integrate in MED

$$J(\lambda) = \sum_t y_t(\lambda'_t - \lambda_t) - \epsilon \sum_t (\lambda'_t + \lambda_t) + \sum_t \log(\lambda_t) - \log\left(1 - e^{-\lambda_t \epsilon} + \frac{\lambda_t}{c - \lambda_t}\right)$$
$$+ \sum_t \log(\lambda'_t) - \log\left(1 - e^{-\lambda'_t \epsilon} + \frac{\lambda'_t}{c - \lambda'_t}\right) - \sum_i \log\left[1 - p_0 + p_0 e^{\frac{1}{2}\left[\sum_t (\lambda_t - \lambda'_t) X_{t,i}\right]^2}\right]$$

- Boston Housing Data: predict price from 13 scalars
  Train/Test=481/25
  Explicit Quadratic Kernel Expansion

Linear Model Estimator	Epsilon-Sensitive Linear Loss
Least-Squares	1.7584
MED p0 = 0.99999	1.7529
MED p0 = 0.1	1.6894
MED p0 = 0.001	1.5377
MED p0 = 0.00001	1.4808



- Cancer Data: predict expression from
  67 other cancer levels
  Train/Test = 50/3951

Linear Model Estimator	Epsilon-Sensitive Linear Loss
Least-Squares	3.609e+03
MED p0 = 0.00001	1.6734e+03

MED Kernel Selection

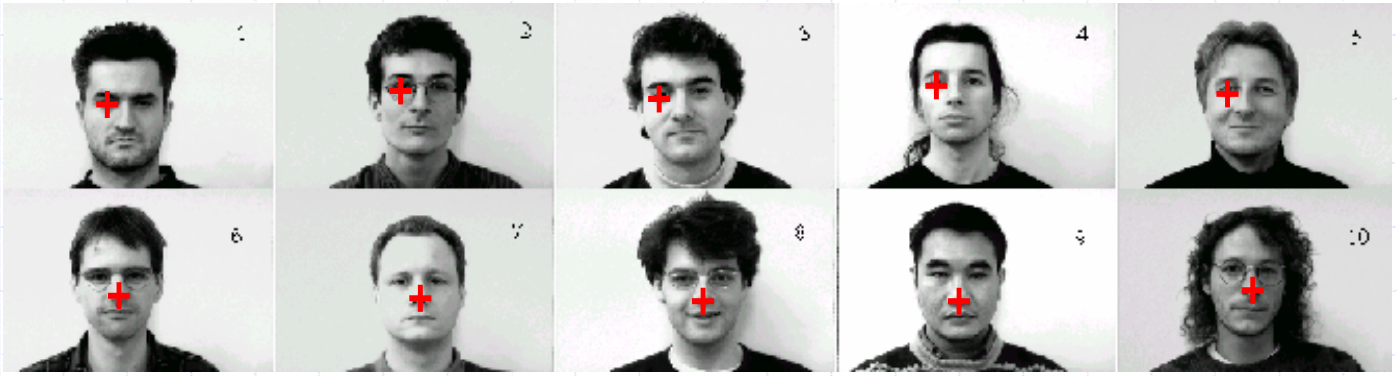
- Purpose: pick mixture of subset of D Kernel matrices to get largest margin classifier, (learn the Gram matrix)
- Turn kernels on/off via binary switches $s_i \in \{0,1\}$
- Switch Prior: Bernoulli distribution $P_{s,0}(s_i) = \rho^{s_i}(1-\rho)^{1-s_i}$
- Discriminant uses D models with multiple nonlinear mappings of datum $L(X;\Theta) = \sum_i s_i \theta_i^T \Phi_i(X) + b$
- MED solution has analytic concave objective fn:

$$J(\lambda) = \sum_t \lambda_t - \sum_{i=1}^{D} \log\left[1-\rho+\rho\exp\left(\tfrac{1}{2}\sum_{t=1}^{T}\sum_{t'=1}^{T}\lambda_t\lambda_{t'}y_t y_{t'}k_i(X_t,X_{t'})\right)\right]$$

With SVM constraints: λ_t in [0,C] and $\Sigma_t \lambda_t y_t = 0$

Meta-Learning

- Learning to Learn: Multi-Task or Meta-Learning
- Use multiple related tasks to improve learning
  typically implemented in Neural Nets (local minima)
  with a shared representation layer and input layer
  (Caruana, Thrun, and Baxter)
- SVMs: typically only find a single classification/regression
- Can we combine multi SVMs for different tasks yet with
  a shared input space and learn a common representation?



Meta Feature Selection

- Given a series of classification tasks: $m \in [1..M]$
  map inputs to binary: $X_{tm} \rightarrow y_{tm} \quad \forall t \in [1..T_m]$
  using M discriminants with 1 feature selection vector:

$$L(X; s, \theta_m, b_m) = \sum_i s_i \theta_{m,i} X_i + b_m$$

Subject to MED classification constraints:

$$\int P(s, \theta_1, \ldots, \theta_M, b_1, \ldots, b_M) \left[y_{tm} \left(L(X_{tm}; s, \theta_m, b_m) - 1 \right) \right] d\Theta \geq 0, \ \forall t \forall m$$

Solve by optimizing joint objective function for all Lagrange

$$J(\lambda) = \sum_{t,m} \lambda_{tm} - \sum_{i=1}^{D} \log \left(1 - \rho + \rho \exp \left(\frac{1}{2} \sum_{m=1}^{M} \left[\sum_{t=1}^{T_m} \lambda_{tm} y_{tm} X_{tm,i} \right]^2 \right) \right)$$

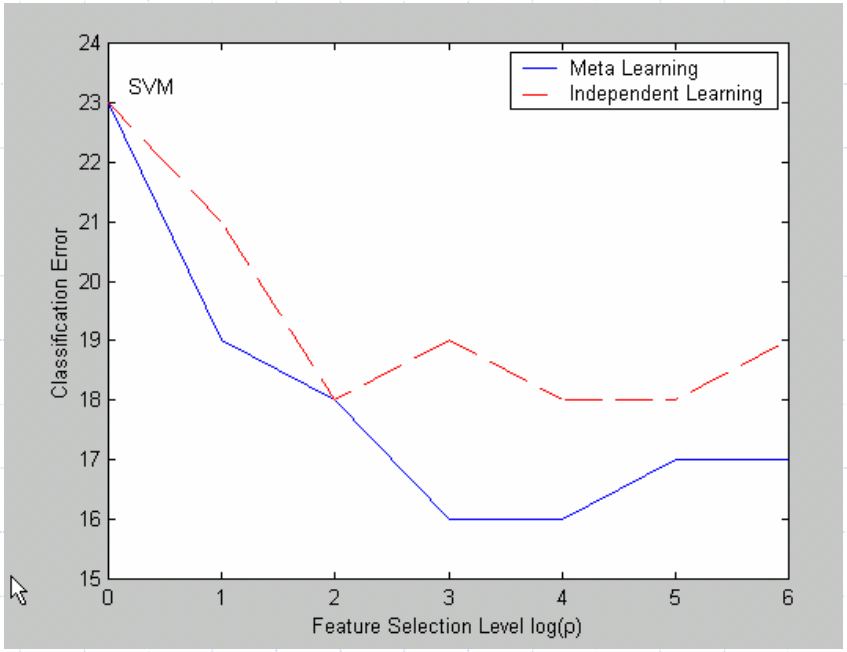
With SVM constraints: λ_{tm} in [0,C] and $\Sigma_t \lambda_{tm} y_{tm} = 0$ for all m

Meta Feature Selection Results

- Have many classification tasks with common feature selection. To ensure coupled tasks, turn multi-class data set into multiple 1 versus many tasks

UCI Dermatology Dataset: 200 trains, 166 tests, 33 features, 6 classes

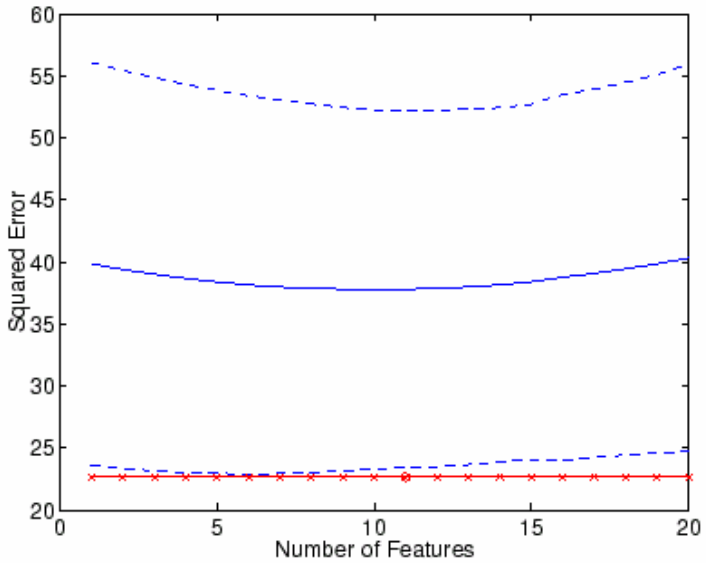
Cross-validating over Regularization Levels

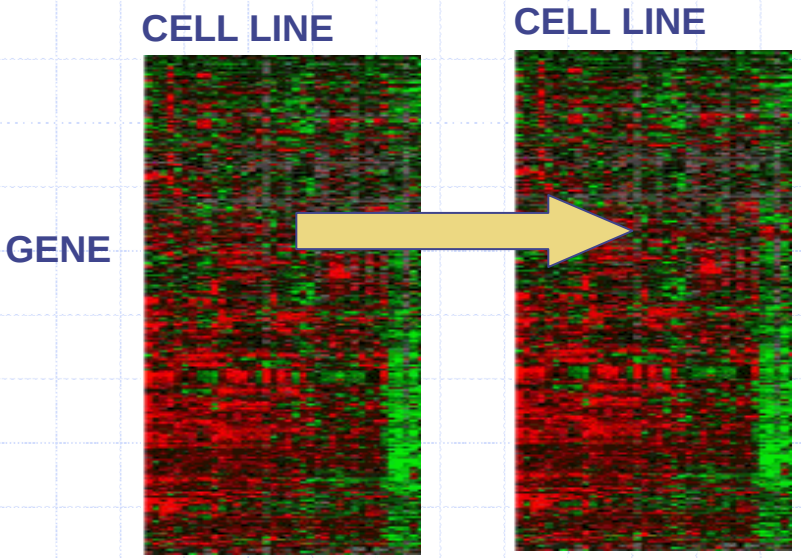


Meta Feature Select Regression

- Given many regression tasks with common feature selection

D. Ross Cancer Data: 67 expression level features.
Use subset of 800 genes to predict all others
Compared with random feature selection





Meta Kernel Selection

- Given many tasks with common (unknown) kernel matrix
- Use M discriminants with 1 feature selection vector:

$$L\left(X;s,\Theta_m,b_m\right)=\sum_i s_i\theta_{m,i}^T\Phi_i\left(X\right)+b_m$$

- Subject to MED classification constraints:

$$\int P\left(s,\Theta_1,\ldots,\Theta_M,b_1,\ldots,b_M\right)\left[y_{tm}L\left(X_{tm};s,\Theta_m,b_m\right)-\gamma\right]dsdb_\forall d\Theta_\forall\geq 0,\ \forall t\forall m$$

optimize joint objective function over Lagrange multipliers

$$J\left(\lambda\right)=\sum_{t,m}\lambda_{tm}-\sum_{i=1}^{D}\log\left[1-\rho+\rho\exp\left(\tfrac{1}{2}\sum_{m=1}^{M}\sum_{t=1}^{T}\sum_{t'=1}^{T}\lambda_{tm}\lambda_{tm'}y_{tm}y_{tm'}k_i\left(X_{tm},X_{tm'}\right)\right)\right]$$

With SVM constraints: λ_{tm} in [0,C] and $\Sigma_t \lambda_{tm}y_{tm}=0$ for all m

Meta Kernel Selection

- Code for learning the weights for d=1…D kernels

Table 1: The Multi-Task SVM Learning Algorithm

0	Input: dataset $\mathcal{D}$, value C and α and kernels k_d for $d = 1, \ldots, D$.
1	Initialize Lagrange multipliers to zero, $\lambda = \vec{0}$ for $m = 1 \ldots M$ and $t = 1, \ldots, T_m$.
2	Store $\tilde{\lambda} = \lambda$.
3	For $m = 1, \ldots, M$ do:
	Set $g_d = \alpha \exp\left(-\frac{1}{2}\sum_{m=1}^{M}\sum_{t=1}^{T_m}\sum_{\tau=1}^{T_m}\lambda_{m,t}\lambda_{m,\tau}y_{m,t}y_{m,\tau}k_d(X_{m,t}, X_{m,\tau})\right)$ for all d. Set $\mathcal{G}_d = \frac{\tanh(\frac{1}{2}\log(g_d))}{2\log(g_d)}$ for all d. Set $\hat{S}(d) = \frac{1}{1+g_d}$ for all d. Set $\hat{Y}_{m,t}(d) = \sum_{\tau=1}^{T_m}\lambda_{m,\tau}y_{m,\tau}k_d(X_{m,t}, X_{m,\tau})$. for all t and d.
	Update λ_m vectors with the SVM QP: $\max_{\lambda_m} \sum_{t=1}^{T_m}\lambda_{m,t} - \sum_{t=1}^{T_m}\lambda_{m,t}y_{m,t}\sum_{d=1}^{D}\hat{S}(d)\hat{Y}_{m,t}(d)$ $+\sum_{t=1}^{T_m}\sum_{\tau=1}^{T_m}\lambda_{m,t}\tilde{\lambda}_{m,\tau}y_{m,t}y_{m,\tau}\sum_{d=1}^{D}\left(\mathcal{G}_d\hat{Y}_{m,t}(d)\hat{Y}_{m,\tau}(d) + k_d(X_{m,t}, X_{m,\tau})\right)$ $-\frac{1}{2}\sum_{t=1}^{T_m}\sum_{\tau=1}^{T_m}\lambda_{m,t}\lambda_{m,\tau}y_{m,t}y_{m,\tau}\sum_{d=1}^{D}\left(\mathcal{G}_d\hat{Y}_{m,t}(d)\hat{Y}_{m,\tau}(d) + k_d(X_{m,t}, X_{m,\tau})\right)$ s.t. $0 \le \lambda_{m,t} \le C \quad \forall t = 1, \ldots, T_m$ s.t. $\sum_{t=1}^{T_m} y_{m,t}\lambda_{m,t} = 0$
4	If $\|\lambda - \tilde{\lambda}\| \ge \epsilon$ go to 2.
5	Output: $\hat{S}$ and λ.

- Final kernel to use in the SVMs: $k\left(X, X'\right) = \sum_{d=1}^{D}\hat{S}\left(d\right)k_d\left(X, X'\right)$

Meta Kernel Selection Results

UCI Isolet data set (letter recognition from audio)
26 Classes used as 1 to Many Binary Classification

200 training
600 testing

SVM (X's)
Kernel Selection (red)
Meta Selection (blue)

Used $\rho = 0.1$
$\rho = 0.01$
$\rho = 0.001$