Advanced Machine Learning & Perception

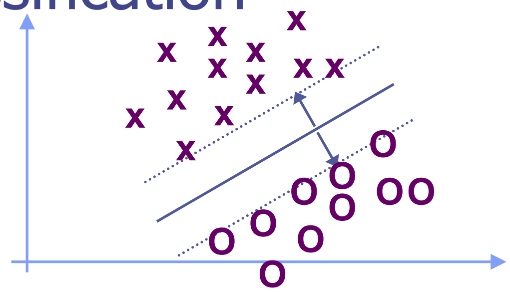
Instructor: Tony Jebara

Topic 6

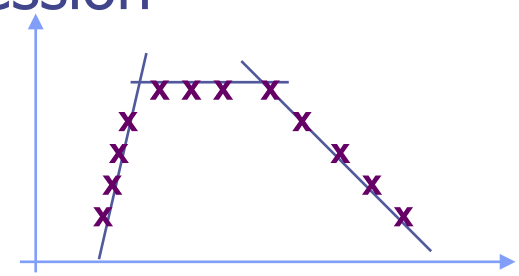
- Review MED SVM
- MED Feature Selection
- MED Kernel Selection
- Multi-Task MED
- Adaptive Pooling

SVM Extensions

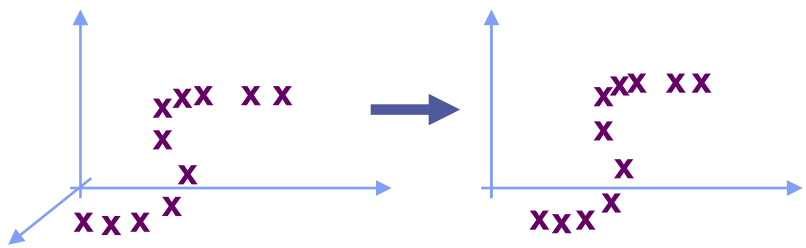
Classification



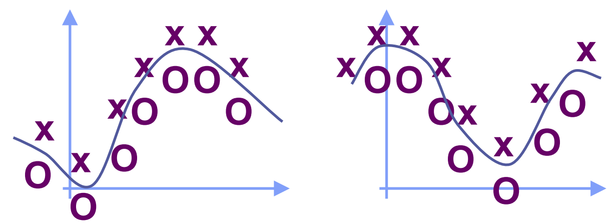
Regression



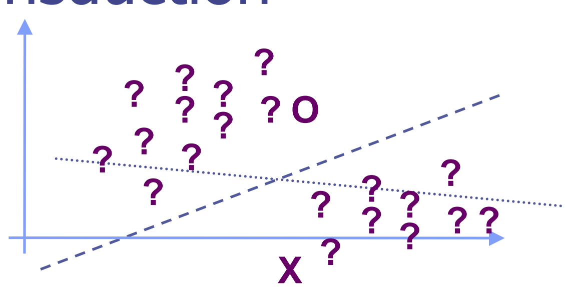
Feature/Kernel Selection



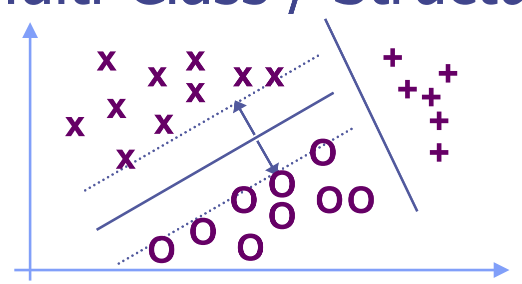
Meta/Multi-Task Learning



Transduction



Multi-Class / Structured



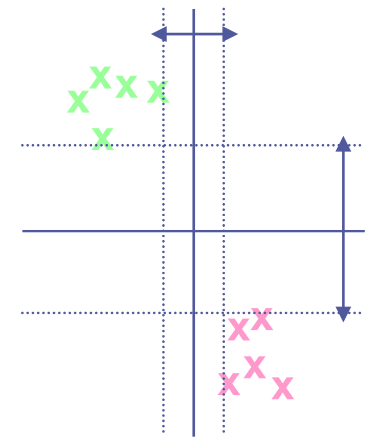
MED Support Vector Machine

- MED approach to find discriminant: $L\left(X;\Theta\right) = \theta^T X + b$
- Get P(θ): $\min_P KL\left(P \| P_0\right) + C\sum_t \xi_t$

  $s.t. \quad \int P\left(\Theta\right) y_t L\left(X_t;\Theta\right) d\Theta \geq 1 - \xi_t, \ \forall t$
- Solution: $P\left(\Theta\right) = \frac{1}{Z\left(\lambda\right)} P_0\left(\Theta\right) \exp\left(\sum_t \lambda_t \left[y_t L\left(X_t;\Theta\right) - 1\right]\right)$
- Partition: $Z\left(\lambda\right) = \int_\Theta P_0\left(\Theta\right) \exp\left(\sum_t \lambda_t \left[y_t L\left(X_t;\Theta\right) - 1\right]\right) d\Theta$
- Objective: (same as SVM) $J\left(\lambda\right) = \max_\lambda \sum_t \lambda_t - \frac{1}{2} \sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} \left(X_t^T X_{t'}\right)$

  $s.t. \ 0 \leq \lambda_t \leq C, \ \sum_t \lambda_t y_t = 0$
- Prediction: $\hat{y} = sgn \int P\left(\Theta\right) L\left(X;\Theta\right) d\Theta = sgn \sum_t y_t \lambda_t X_t^T X + b$

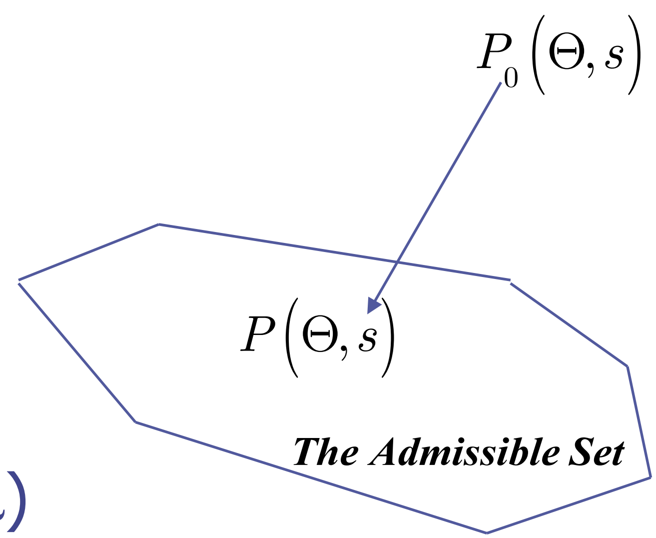
MED Feature Selection



- Goal: pick 100 of 10000 features to get largest margin classifier (NP)
- Turn features on/off via binary switches $s_i \in \{0,1\}$
- Discriminant is now $L(X;\Theta) = \sum_i s_i \theta_i X_i + b$
- Introduce a prior on switches:

$$P_{s,0}(s_i) = \rho^{s_i}(1-\rho)^{1-s_i}$$

- This is a Bernoulli distribution where ρ controls a priori pruning level
- MED finds discriminative P(θ,s) close to prior by maximizing J(λ)= –log Z(λ)

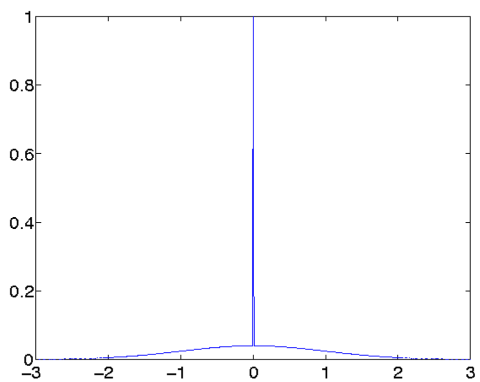


MED Feature Selection

- Discriminant function now is $L\left(X;\Theta\right)=\sum_{i=1}^{D}s_i\theta_i X_i+b$
- The model $\Theta=\left\{b,\theta_1,\ldots,\theta_D,s_1,\ldots,s_D\right\}$ contains binary $s_i\in\left\{0,1\right\}$ parameters (with Bernouilli priors) to prune features

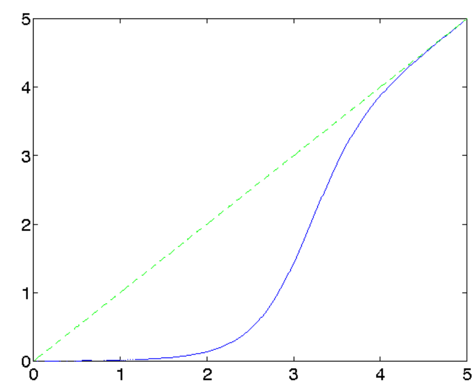
Prior: $P_0\left(\Theta\right)=P_0\left(b\right)P_0\left(\theta\right)P_0\left(s\right)=N\left(b\mid 0,\infty\right)N\left(\theta\mid 0,I\right)\prod_i P_0\left(s_i\right)$

Partition: $Z\left(\lambda\right)=\sum_{s_1=0}^{1}\cdots\sum_{s_D=0}^{1}\int_b\int_\theta P_0\left(\Theta\right)\exp\left(\lambda_t\left[y_t L\left(X_t;\Theta\right)-1\right]\right)d\theta\, db$



Prior on $s_i\theta_i$

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

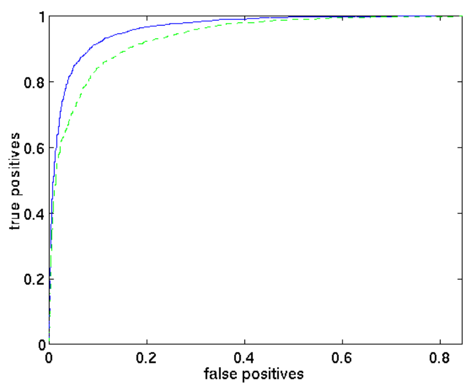


MED Feature Selection

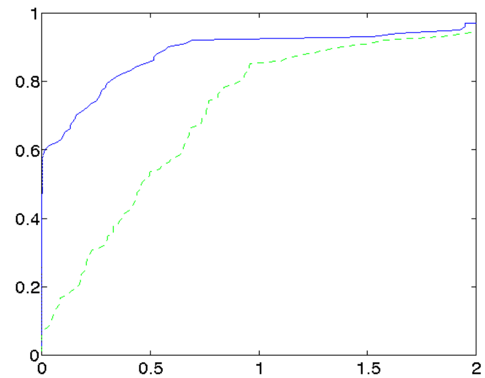
Objective is now: $$J(\lambda) = \sum_t \lambda_t - \sum_{i=1}^{D} \log\left[1 - \rho + \rho e^{\frac{1}{2}\left(\sum_t \lambda_t y_t X_{t,i}\right)^2}\right]$$

$$s.t.\ \ 0 \leq \lambda_t \leq C,\ \ \sum_t \lambda_t y_t = 0$$

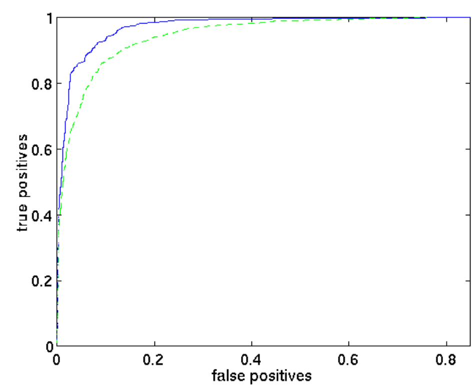
DNA Data: 2-class, 100 element binary vectors. Train/Test=500/4724



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: $\rho = 0.99999$ Solid line: $\rho = 0.00001$

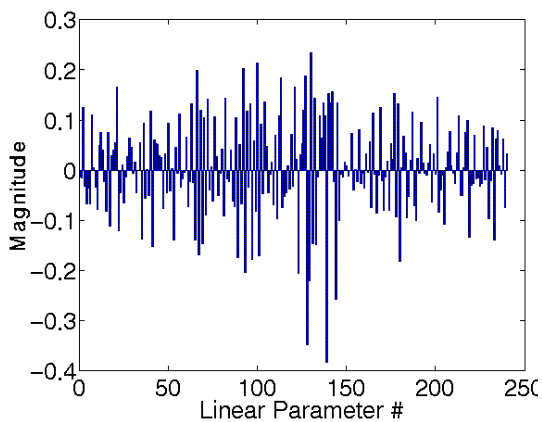
MED Feature Selection

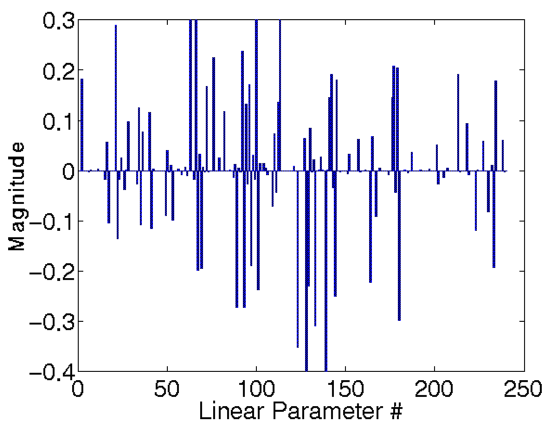
$$J(\lambda) = \sum_t \lambda_t - \sum_{i=1}^{D} \log\left[1 - \rho + \rho e^{\frac{1}{2}\left(\sum_t \lambda_t y_t X_{t,i}\right)^2}\right]$$

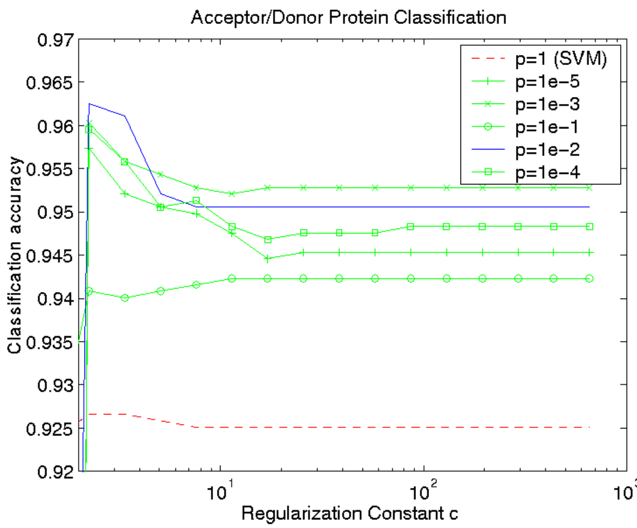
$$s.t.\ \ 0 \leq \lambda_t \leq C,\ \ \sum_t \lambda_t y_t = 0$$

Example: Intron-Exon Protein Classification:

UCI: 240 dims; 200 train, 1300 test







MED Feature Selection

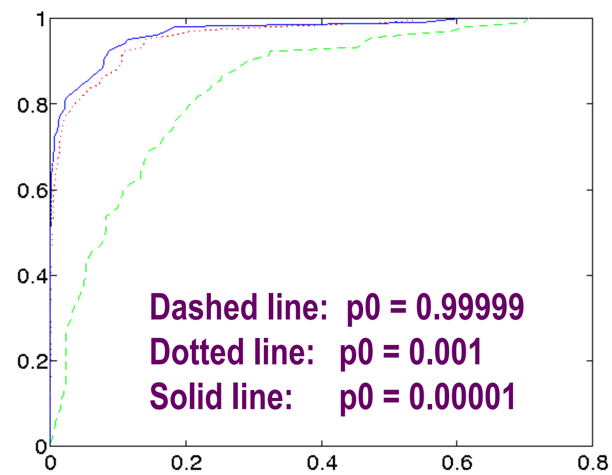
- MED can also use switches in regression, objective is then:

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

$$s.t. 0 \leq \lambda'_t, \lambda_t \leq C, \; \sum_t \lambda'_t - \lambda_t = 0$$

- 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

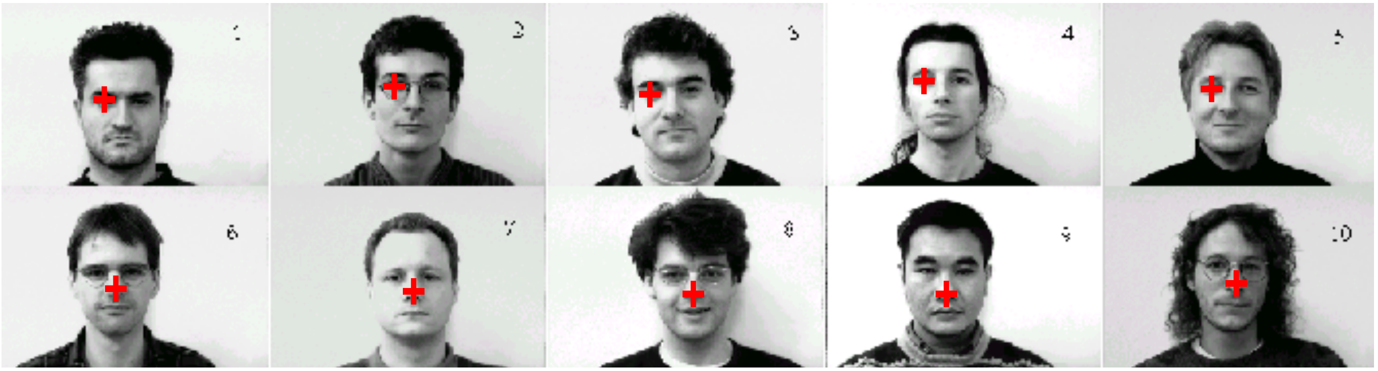
- Purpose: pick mixture of subset of D Kernel matrices to get largest margin classifier (i.e. learn the Gram matrix)
- Turn kernels on/off via binary switches $s_i \in \{0,1\}$
- Switch Prior: Bernoulli distribution $P_{s,0}(s_i) = \rho^{s_i}(1-\rho)^{1-s_i}$
- Discriminant uses D models with multiple 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:

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

$$s.t.\ \ 0 \leq \lambda_t \leq C,\ \ \sum_t \lambda_t y_t = 0$$

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: $m \in \left[1..M\right]$

  map inputs to binary: $X_{tm} \rightarrow y_{tm} \qquad \forall t \in \left[1..T_m\right]$

  using M discriminants with 1 feature selection vector:

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

Subject to MED classification constraints:

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

Solve by optimizing joint objective function for all Lagranges

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

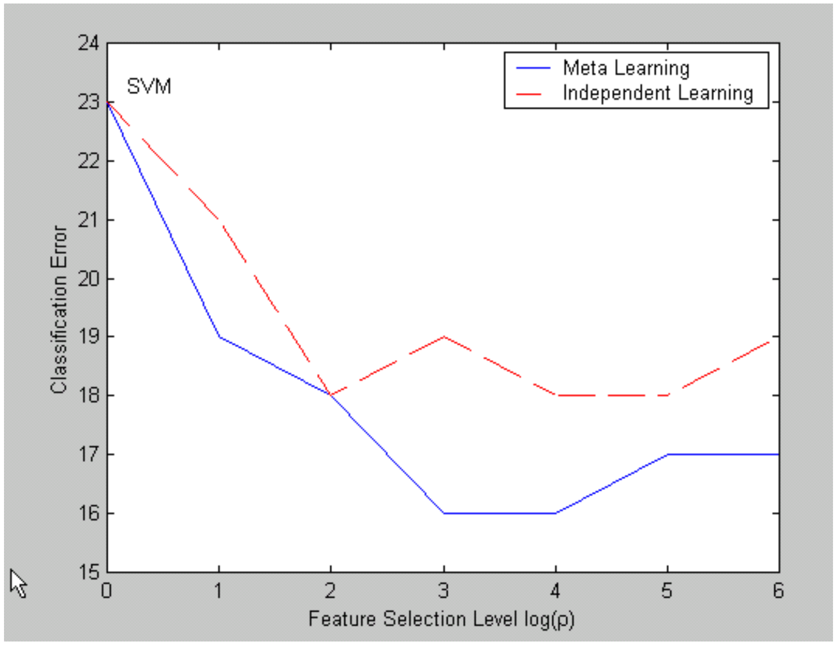
$$s.t. \ \ 0 \leq \lambda_{tm} \leq C, \ \ \sum_t \lambda_{tm} y_{tm} = 0 \quad \forall m$$

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

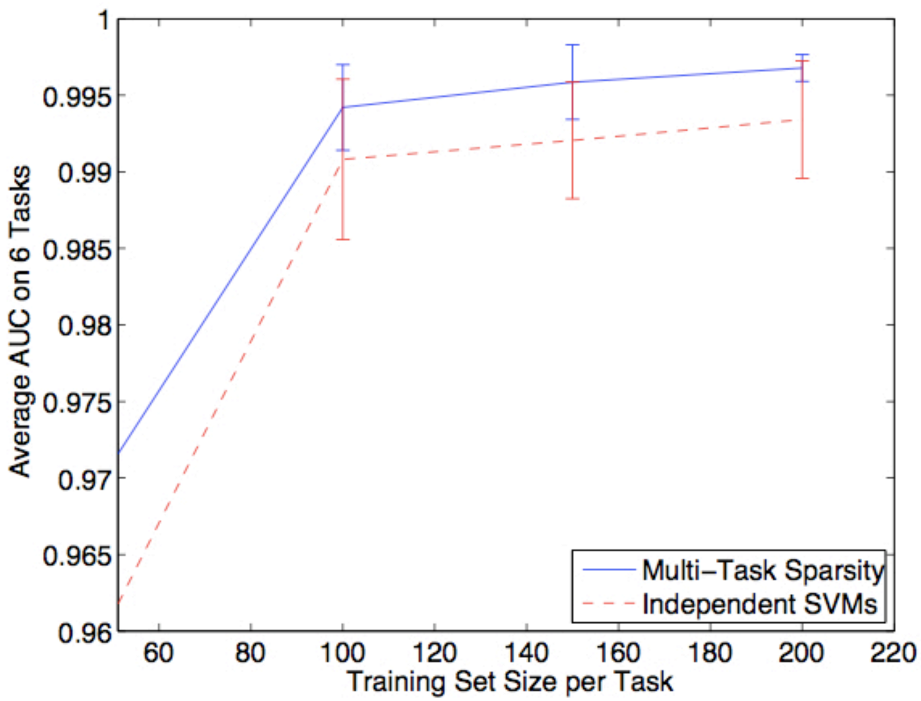
Cross-validating over Regularization Levels



Meta Feature Selection Results

Can also cross validate over ρ (or $\alpha=(1-\rho)/\rho$) as well as C

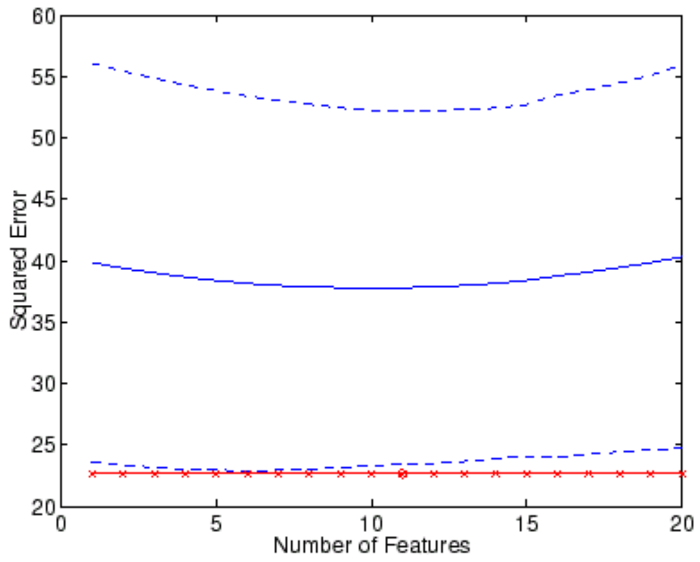
Example: UCI Dermatology dataset (6 tasks)

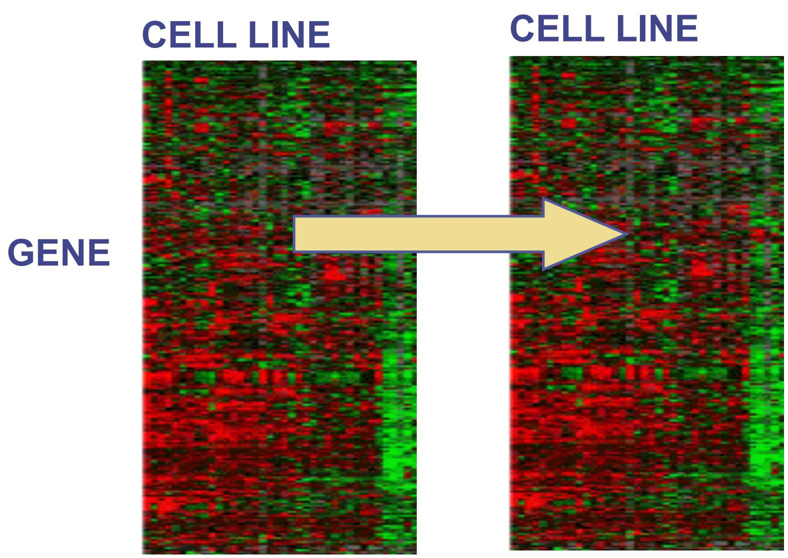


Meta Feature Select Regression

- Can also solve many regression tasks with one 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 one 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:

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

optimize joint objective function over Lagrange multipliers

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

$$s.t. \;\; 0 \leq \lambda_{tm} \leq C, \;\; \sum_t \lambda_{tm} y_{tm} = 0 \quad \forall m$$

Meta Kernel Selection as QP

- The objective function is convex but not quite a QP

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

$$s.t.\;\; 0 \le \lambda_{tm} \le C, \;\; \sum_t \lambda_{tm} y_{tm} = 0 \quad \forall m$$

- Use a bound on each log term to make it quadratic in λ

$$-\log\left(\alpha + \exp\left(\frac{\mathbf{u}^T\mathbf{u}}{2}\right)\right) \ge -\log\left(\alpha + \exp\left(\frac{\mathbf{v}^T\mathbf{v}}{2}\right)\right) - \frac{\exp\left(\frac{\mathbf{v}^T\mathbf{v}}{2}\right)}{\alpha + \exp\left(\frac{\mathbf{v}^T\mathbf{v}}{2}\right)} \mathbf{v}^T\left(\mathbf{u} - \mathbf{v}\right) - \frac{1}{2}\left(\mathbf{u} - \mathbf{v}\right)^T \left(\mathfrak{G}\mathbf{v}^T\mathbf{v} + I\right)\left(\mathbf{u} - \mathbf{v}\right)$$

$$where\;\; \mathfrak{G} = \frac{\tanh\left(\tfrac{1}{2}\log\left(\alpha \exp\left(-\mathbf{v}^T\mathbf{v}/2\right)\right)\right)}{2\log\left(\alpha \exp\left(-\mathbf{v}^T\mathbf{v}/2\right)\right)}$$

- As with EM, maximize the lower bound, update & repeat

- Converges in fewer steps than $\left\lceil \frac{\log\left(1/\varepsilon\right)}{\log\left(\min\left(1 + 1/\alpha, 2\right)\right)} \right\rceil$

Meta Kernel Selection as QP

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

Algorithm 1 Multitask SVM Learning

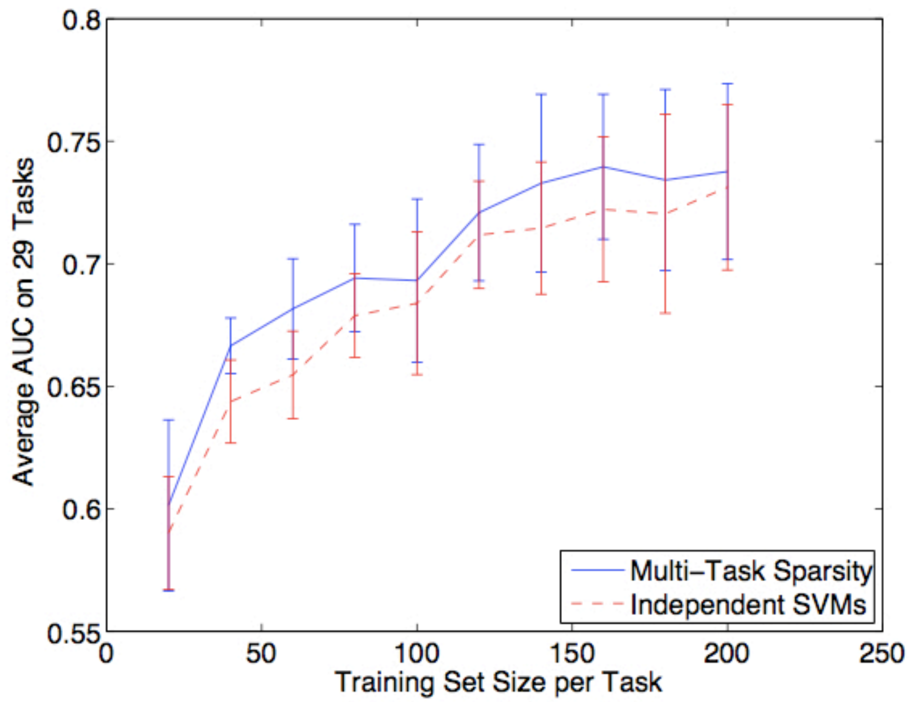
0	Input dataset $\mathcal{D}$, $C>0$, $\alpha \geq 0$, $0<\varpi<1$ and kernels k_d for $d=1,\ldots,D$.
1	Initialize Lagrange multipliers to zero $\boldsymbol{\lambda}=\mathbf{0}$.
2	Store $\tilde{\boldsymbol{\lambda}}=\boldsymbol{\lambda}$.
3	For $m=1,\ldots,M$ do:
3a	Set $g_d=\alpha\exp\left(-\frac{1}{2}\sum_{m=1}^{M}\sum_{t=1}^{T_m}\sum_{\tau=1}^{T_m}\lambda_{m,t}\lambda_{m,\tau}y_{m,t}y_{m,\tau}k_d(\mathbf{x}_{m,t},\mathbf{x}_{m,\tau})\right)$ for all d. Set $\mathcal{G}_d=\frac{\tanh(\frac{1}{2}\log(g_d))}{2\log(g_d)}$ for all d. Set $\hat{\mathbf{s}}(d)=\frac{1}{1+g_d}$ for all d. Set $\hat{\mathbf{y}}_{m,t}(d)=\sum_{\tau=1}^{T_m}\lambda_{m,\tau}y_{m,\tau}k_d(\mathbf{x}_{m,t},\mathbf{x}_{m,\tau})$ for all t and d.
3b	Update each of the $\boldsymbol{\lambda}_m$ vectors with the SVM QP: $\max_{\boldsymbol{\lambda}_m}\sum_{t=1}^{T_m}\lambda_{m,t}-\sum_{t=1}^{T_m}\lambda_{m,t}y_{m,t}\sum_{d=1}^{D}\hat{\mathbf{s}}(d)\hat{\mathbf{y}}_{m,t}(d)$ $+\sum_{t=1}^{T_m}\sum_{\tau=1}^{T_m}\lambda_{m,t}\tilde{\lambda}_{m,\tau}y_{m,t}y_{m,\tau}\sum_{d=1}^{D}(\mathcal{G}_d\hat{\mathbf{y}}_{m,t}(d)\hat{\mathbf{y}}_{m,\tau}(d)+k_d(\mathbf{x}_{m,t},\mathbf{x}_{m,\tau}))$ $-\frac{1}{2}\sum_{t=1}^{T_m}\sum_{\tau=1}^{T_m}\lambda_{m,t}\lambda_{m,\tau}y_{m,t}y_{m,\tau}\sum_{d=1}^{D}(\mathcal{G}_d\hat{\mathbf{y}}_{m,t}(d)\hat{\mathbf{y}}_{m,\tau}(d)+k_d(\mathbf{x}_{m,t},\mathbf{x}_{m,\tau}))$ s.t. $0\leq\lambda_{m,t}\leq C\ \ \forall t=1,\ldots,T_m$ and $\sum_{t=1}^{T_m}y_{m,t}\lambda_{m,t}=0$.
4	If $\|\boldsymbol{\lambda}-\tilde{\boldsymbol{\lambda}}\|>\varpi\|\boldsymbol{\lambda}\|$ go to 2.
5	Output: $\hat{\mathbf{s}}$ and $\boldsymbol{\lambda}$.

- 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

Can also cross validate over ρ (or $\alpha=(1-\rho)/\rho$) as well as C

Example: Landmine dataset (29 tasks) with RBF kernels



Meta or Adaptive Pooling

- Another type of meta-learning is adaptive pooling
- Assume $m \in [1..M]$ datasets predicting binary labels
- Here, datasets are all labeled for the same task
- But, inputs are sampled from slightly different distributions
- E.g. Dataset 1: color face images labeled as male/female
  Dataset 2: gray face images labeled as male/female
- Pooling: combine both datasets and learn one classifier

$$L_m(X) = \theta^T X + b$$

- Independent learning: learn a separate classifier for each

$$L_m(X) = \theta_m^T X + b_m$$

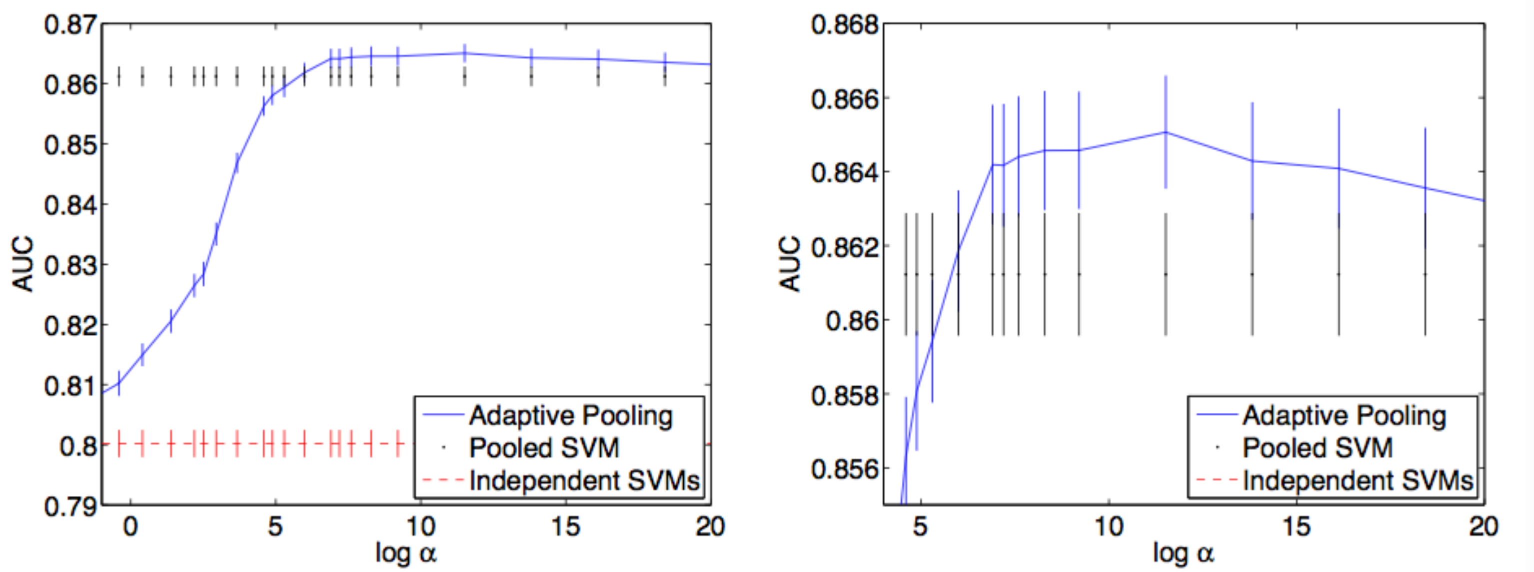
- Adaptive pooling: each classifier is a mix of the shared model and a specialized model

$$L_m(X) = s_m\left(\theta_m^T X + b_m\right) + \left(\theta^T X + b\right)$$

- Once again MED solution is straightforward…

Meta or Adaptive Pooling

- Compare to full pooling and independent learning



(a) Average AUC (b) Average AUC zoomed in

$$J(\lambda) = \sum_{t,m} \lambda_{tm} - \sum_{m} \sum_{m'} \frac{1}{2} \sum_{t=1}^{T_m} \sum_{t'=1}^{T_m} \lambda_{tm} \lambda_{t'm'} y_{tm} y_{t'm'} k\left(X_{tm}, X_{t'm'}\right)$$

$$\sum_{m=1}^{M} \log \left[\alpha + \exp\left(\frac{1}{2} \sum_{t=1}^{T_m} \sum_{t'=1}^{T_m} \lambda_{tm} \lambda_{tm'} y_{tm} y_{tm'} k_m \left(X_{tm}, X_{tm'}\right) \right) \right] + M \log(\alpha + 1)$$

$$s.t. \;\; 0 \le \lambda_{tm} \le C, \;\; \sum_{t} \lambda_{tm} y_{tm} = 0 \quad \forall m$$