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: \( L(X; \Theta) = \theta^T X + b \)

• Get \( P(\theta) \): 
  \[
  \min_P KL(P \parallel P_0) + C \sum_t \xi_t \\
  \text{s.t. } \int P(\Theta) y_t L(X_t; \Theta) d\Theta \geq 1 - \xi_t, \forall t
  \]

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

• Partition: 
  \[
  Z(\lambda) = \int_{\Theta} P_0(\Theta) \exp \left( \sum_t \lambda_t \left[ y_t L(X_t; \Theta) - 1 \right] \right) d\Theta
  \]

• Objective: 
  \[
  J(\lambda) = \max_\lambda \sum_t \lambda_t - \frac{1}{2} \sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} (X_t^T X_{t'}) \\
  \text{s.t. } 0 \leq \lambda_t \leq C, \ \sum_t \lambda_t y_t = 0
  \]

• Prediction: 
  \[
  \hat{y} = sgn \int P(\Theta) L(X; \Theta) d\Theta = sgn \sum_t y_t \lambda_t X_t^T X + b
  \]
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:

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

• This is a Bernoulli distribution where \( \rho \) controls a priori pruning level

• MED finds discriminative \( P(\theta, s) \) close to prior by maximizing \( J(\lambda) = -\log Z(\lambda) \)
MED Feature Selection

• Discriminant function now is
  \[ L(X; \Theta) = \sum_{i=1}^{D} s_i \theta_i X_i + b \]

• The model \( \Theta = \{b, \theta_1, \ldots, \theta_D, s_1, \ldots, s_D\} \) contains binary \( s_i \in \{0,1\} \) parameters (with Bernouilli priors) to prune features

Prior:
\[ P_0(\Theta) = P_0(b) P_0(\theta) P_0(s) = N(b | 0, \infty) N(\theta | 0, I) \prod_i P_0(s_i) \]

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

Prior on \( s_i \theta_i \)

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

Objective is now:

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

\[
s.t. \quad 0 \leq \lambda_t \leq C, \quad \sum_t \lambda_t y_t = 0
\]

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

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

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:
  \[ J(\lambda) = \sum_{t} y_t (\lambda'_t - \lambda_t) - \epsilon \sum_{t} (\lambda'_t + \lambda_t) - \sum_i \log \left( 1 - p_0 + p_0 e^{\frac{1}{2}[\sum_t (\lambda_t' - \lambda_t) x_{t,i}]} \right) \]
  \[ s.t. \ 0 \leq \lambda'_t, \lambda_t \leq C, \ \sum_t \lambda'_t - \lambda_t = 0 \]

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

<table>
<thead>
<tr>
<th>Linear Model Estimator</th>
<th>Epsilon-Sensitive Linear Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Least-Squares</td>
<td>1.7584</td>
</tr>
<tr>
<td>MED p0 = 0.99999</td>
<td>1.7529</td>
</tr>
<tr>
<td>MED p0 = 0.1</td>
<td>1.6894</td>
</tr>
<tr>
<td>MED p0 = 0.001</td>
<td>1.5377</td>
</tr>
<tr>
<td>MED p0 = 0.00001</td>
<td>1.4808</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Linear Model Estimator</th>
<th>Epsilon-Sensitive Linear Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Least-Squares</td>
<td>3.6099e+03</td>
</tr>
<tr>
<td>MED p0 = 0.00001</td>
<td>1.6734e+03</td>
</tr>
</tbody>
</table>
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:

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

\[s.t. \ 0 \leq \lambda_t \leq C, \ \sum_t \lambda_t y_t = 0\]
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: $m \in [1..M]$
  
  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:

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

Solve by optimizing joint objective function for all Lagranges

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

s.t. $0 \leq \lambda_{tm} \leq C, \quad \sum_{t} \lambda_{tm} y_{tm} = 0 \quad \forall m$
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 $\rho$ (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:
  \[ L(X; s, \Theta_m, b_m) = \sum_i s_i \theta_{m,i}^T \Phi_i(X) + b_m \]
- Subject to MED classification constraints:
  \[
  \int P(s, \Theta_1, ..., \Theta_M, b_1, ..., b_M) \left[ y_{tm} L(X_{tm}; s, \Theta_m, b_m) - \gamma \right] ds db \, d\Theta_m \geq 0, \forall t \forall m
  \]

Optimize joint objective function over Lagrange multipliers

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

s.t. \[ 0 \leq \lambda_{tm} \leq C, \sum_t \lambda_{tm} y_{tm} = 0 \, \forall m \]
Meta Kernel Selection as QP

• The objective function is convex but not quite a QP

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

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

• Use a bound on each log term to make it quadratic in \( \lambda \)

\[ -\log \left( \alpha + \exp \left( \frac{u^T u}{2} \right) \right) \geq -\log \left( \alpha + \exp \left( \frac{v^T v}{2} \right) \right) - \frac{\exp \left( \frac{v^T v}{2} \right)}{\alpha + \exp \left( \frac{v^T v}{2} \right)} v^T (u - v) - \frac{1}{2} (u - v)^T (G v^T v + I) (u - v) \]

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

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

• Converges in fewer steps than

\[ \left[ \frac{\log (1 / \varepsilon)}{\log \left( \min \left( 1 + 1 / \alpha, 2 \right) \right)} \right] \]
Meta Kernel Selection as QP

- Code for learning the weights for $d=1...D$ kernels

**Algorithm 1 Multitask SVM Learning**

0. Input dataset $D, C > 0, \alpha \geq 0, 0 < \omega < 1$ and kernels $k_d$ for $d = 1, \ldots, D$.

1. Initialize Lagrange multipliers to zero $\lambda = 0$.

2. Store $\tilde{\lambda} = \lambda$.

3. For $m = 1, \ldots, M$ do:
   3a. Set $g_d = \alpha \exp\left(-\frac{1}{2} \sum_{m=1}^{M} \sum_{t=1}^{T_m} \lambda_{m,t} \lambda_{m,\tau} y_m \tau y_m \tau k_d(x_m,t, x_m,\tau)\right)$ for all $d$.

   Set $G_d = \frac{\tanh(\frac{1}{2} \log(g_d))}{2 \log(g_d)}$ 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 $d$.

3b. Update each of the $\lambda_{m,t}$ vectors with the SVM QP:

\[
\max_{\lambda_m, \lambda_{m,t}, \tau} \sum_{t=1}^{T_m} \lambda_{m,t} - \sum_{t=1}^{T_m} \lambda_{m,\tau} y_m \tau \sum_{d=1}^{D} \hat{s}(d) \hat{y}_{m,t}(d)
\]

\[
+ \sum_{t=1}^{T_m} \lambda_{m,t} \hat{y}_{m,\tau}(d) y_m \tau \sum_{d=1}^{D} (G_d \hat{y}_{m,t}(d) \hat{y}_{m,\tau}(d) + k_d(x_m,t, x_m,\tau))
\]

\[
- \frac{1}{2} \sum_{t=1}^{T_m} \sum_{\tau=1}^{T_m} \lambda_{m,t} \lambda_{m,\tau} y_m \tau y_m \tau \sum_{d=1}^{D} (G_d \hat{y}_{m,t}(d) \hat{y}_{m,\tau}(d) + k_d(x_m,t, x_m,\tau))
\]

s.t. $0 \leq \lambda_{m,t} \leq C \quad \forall t = 1, \ldots, T_m$ and $\sum_{t=1}^{T_m} y_m \tau \lambda_{m,\tau} = 0$.

4. If $\|\lambda - \tilde{\lambda}\| > \omega \|\lambda\|$ go to 2.

5. Output: $\hat{s}$ and $\lambda$.

**Final kernel to use in the SVMs:**

\[
k(X, X') = \sum_{d=1}^{D} \hat{S}(d) k_d(X, X')
\]
Meta Kernel Selection Results

Can also cross validate over $\rho$ (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

\[
J(\lambda) = \sum_{t,m} \lambda_{tm} - \sum_m \sum_{m'} \frac{1}{2} \sum_{t=1}^{T} \sum_{t'=1}^{T} \lambda_{tm} \lambda_{t'm'} y_{tm} y_{t'm'} k_m(X_{tm}, X_{t'm'})
\]

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

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