The key here is to derive the equivalent reformulation for the problem

$$\max_{\|\mathbf{u}\|\leq\mathbf{1}} \boldsymbol{x}^T\boldsymbol{\Delta}\boldsymbol{u}. \tag{*}$$

Notice from definition $\|u\| = \max\{\frac{1}{\Gamma}\|\boldsymbol{u}\|_1, \|\boldsymbol{u}\|_\infty\}$, therefore $\|\boldsymbol{u}\| \leq \mathbf{1}$ can be equivalently reformulated as $\frac{1}{\Gamma}\|\boldsymbol{u}\|_1 \leq \mathbf{1}, \|\boldsymbol{u}\|_\infty \leq \mathbf{1}$. So we know (*) has an equivalent LO formulation as follows.

$$\begin{aligned} \max\ & (\boldsymbol{\Delta}^T x)^T\boldsymbol{u} \\ \text{s.t.}\ & \sum_i u_i \leq \Gamma \\ & \mathbf{0} \leq \boldsymbol{u} \leq \mathbf{1}. \end{aligned} \tag{P}$$

And its dual problem as:

$$\begin{aligned} \min\ & \Gamma p + \mathbf{1}^T\boldsymbol{q} \\ \text{s.t.}\ & p\mathbf{1} + \boldsymbol{q} \geq |\boldsymbol{\Delta}^T\boldsymbol{x}| \\ & p \geq 0, \boldsymbol{q} \geq \mathbf{0}. \end{aligned} \tag{D}$$

So we know the robust counterpart of the original RO problem is

$$\begin{aligned} \max\ & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.}\ & \hat{\boldsymbol{a}}^T\boldsymbol{x} + \Gamma p + \mathbf{1}^T\boldsymbol{q} \leq b \\ & p\mathbf{1} + \boldsymbol{q} \geq \boldsymbol{\Delta}^T\boldsymbol{x} \\ & p\mathbf{1} + \boldsymbol{q} \geq -\boldsymbol{\Delta}^T\boldsymbol{x} \\ & p \geq 0, \boldsymbol{q} \geq \mathbf{0}. \end{aligned}$$

(a) Note

$$\max_{S\subset N,|S|\leq\Gamma_1} \sum_{j\in S} d_j x_j = \max \sum_{j=1}^{n} d_j x_j z_{1j} \qquad (P_S)$$
$$\text{s.t. } \sum_{j=1}^{n} z_{1_j} \leq \Gamma_1$$
$$0 \leq z_{1j} \leq 1, \forall\ j = 1,\ldots,n.$$

The dual problem of (P_S) is formulated as follows.

$$\min\ \Gamma_1 z_1 + \sum_{j=1}^{n} p_j \qquad (D_S)$$
$$\text{s.t. } p_j + z_1 \geq d_j x_j, \forall\ j = 1,\ldots,n$$
$$z \geq 0, \boldsymbol{p} \geq 0.$$

Similarly, we can get almost identical dual formulation (D_T) of problem $\max_{T\subset M,|T|\leq\Gamma_2} \sum_{k\in T} f_k x_k$. And note these two problems are in fact separable in the original combinatorial optimization. So we arrive at the robust counterpart as

$$\min\ \boldsymbol{c}^T\boldsymbol{x} + \sum_{j=1}^{n} p_j + \sum_{k=n+1}^{2n} p_k + \Gamma_1 z_1 + \Gamma_2 z_2 \qquad (*)$$
$$p_j + z_1 \geq d_j x_j, \forall\ j = 1,\ldots,n$$
$$p_k + z_2 \geq f_k x_k, \forall\ k = n+1,\ldots,2n$$
$$z_1 \geq 0, z_2 \geq 0, \boldsymbol{x} \in X \subset \{0,1\}^{2n}.$$

(b) Note since $\boldsymbol{x}$ is a binary vector, it is clear that z_1 can only take values in $\{0, d_1, \ldots, d_n\}$ and z_2 can only take values in $\{0, f_1, \ldots, f_n\}$. And once z_1, z_2 are fixed, the values of p_j, p_k are in fact also fixed to be $d_j x_j - z_1$ and $f_k x_k - z_2$. So, the problem reduces to $(n+1)^2$ linear binary subprograms for which we have fast subroutines to solve. The optimal solution is just the solution $\boldsymbol{x}^*$ with the least objective value among the $(n+1)^2$ solutions.

The empirical performance of RO in classification using the Framingham Heart Study dataset is studied in this section. To be specific, we compare the performance of the nonregularized SVM and the robust SVM. We follow [1] for the definition of the robust SVM and we discuss robustness in feature and label separately.

The feature-robustness is defined by the uncertainty set (for simplicity, we use the L_1 norm so the robust problem will be linear)

$$\mathcal{U}_x = \{\boldsymbol{\Delta X} \in \mathbb{R}^{n\times d} : \|\boldsymbol{\Delta} x_i\|_1 \leq \Gamma, i = 1, \ldots, n\}$$

and the robust problem is (data vary in features)

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\Delta X}\in\mathcal{U}_x} \sum_{i=1}^{n} \max\{1 - y_i(\boldsymbol{w}^T(x_i + \boldsymbol{\Delta} x_i) - b), 0\} \tag{rSVMf}$$

It can be shown the robust counterpart of (rSVMf) is

$$\begin{aligned}
\min\ & \sum_{i=1}^{n} c_i z_i \\
\text{s.t. } & z_i \geq 1 - y_i(\boldsymbol{w}^T(x_i + \boldsymbol{\Delta} x_i) - b) + \Gamma\|\boldsymbol{w}\|_\infty, \forall\ i = 1, \ldots, n \\
& \boldsymbol{z} \geq \mathbf{0}.
\end{aligned}$$

The label robustness is defined as

$$\mathcal{U}_y = \left\{ \Delta y \in \{0,1\}^n : \sum_{i=1}^{n} \Delta y_i \leq \Gamma \right\}$$

and the robust problem is (data vary in labels)

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\Delta y}\in\mathcal{U}_y} \sum_{i=1}^{n} \max\{1 - y_i(1 - 2\Delta y_i)(\boldsymbol{w}^T(x_i + \boldsymbol{\Delta} x_i) - b), 0\} \tag{rSVMl}$$

and it can be shown the robust counterpart of (rSVMl) is (M is a big constant)

$$\begin{aligned}
\min\ & \sum_{i=1}^{n} c_i z_i + \Gamma q + \sum_{i=1}^{n} r_i \\
\text{s.t. } & q + r_i \geq c_i(f_i - z_i)\ \forall\ i = 1, \ldots, n \\
& z_i \geq 1 - y_i(\boldsymbol{w}^T x_i - b)\ \forall\ i = 1, \ldots, n \\
& z_i \leq 1 - y_i(\boldsymbol{w}^T x_i - b) + M(1 - s_i)\ \forall\ i = 1, \ldots, n \\
& z_i \leq M s_i\ \forall\ i = 1, \ldots, n \\
& f_i \geq 1 + y_i(\boldsymbol{w}^T x_i - b)\ \forall\ i = 1, \ldots, n \\
& f_i \leq 1 + y_i(\boldsymbol{w}^T x_i - b) + M(1 - t_i)\ \forall\ i = 1, \ldots, n \\
& f_i \leq M t_i\ \forall\ i = 1, \ldots, n \\
& r_i, z_i, f_i \geq 0\ \forall\ i = 1, \ldots, n \\
& q \geq 0 \\
& \boldsymbol{s}, \boldsymbol{t} \in \{0,1\}^n.
\end{aligned}$$

We split the dataset into training, validation and test datasets using the 40/40/20 ratio. We test the tree algorithms, the nominal SVM, rSVMf, rSVMl on the training dataset and report the respective test error rates for different Γ in Figure 1, from which we see both rSVMf and rSVMl show better performance than the nominal SVM on the test dataset, with Γ suitably chosen. According to our validation result, the best Γ for rSVMf is 20 and for rSVMl 30.

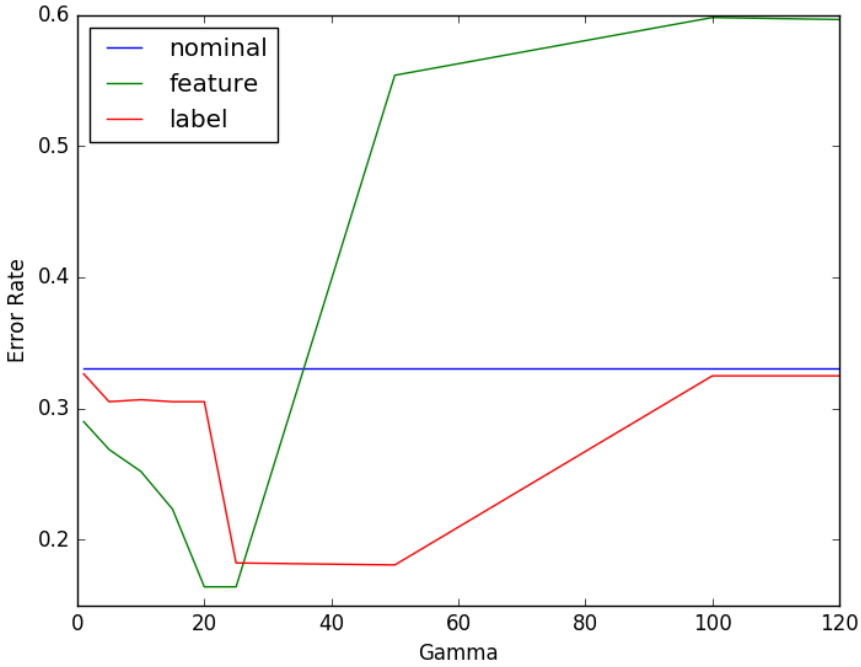

Figure 1: Test error rates of the nominal SVM, rSVMf, rSVMl.

References

[1] D. Bertsimas, J. Dunn, C. Pawlowski, Y. Zhuo. "Robust Classification". *Submitted*, Nov 2015.

Pset2_JuliaCode

March 7, 2017

In [1]:
```
using JuMP, Gurobi, PyPlot
```

```
INFO: Precompiling module JuMP.
```

In [27]:
```
X = readcsv("Framingham.csv");
X_train = X[2:1500,1:15];
y_train = X[2:1500,16]
X_validate = X[1501:3001,1:15];
y_validate = X[1501:3001,16]
X_test = X[3001:3659,1:15];
y_test = X[3001:3659,16];
```

In [22]:
```
n = 3658
d = 15
cost_vector = zeros(n,1)
for i = 1:n
    if X[i+1,d+1] == -1
        cost_vector[i] = 1
    else
        cost_vector[i] = 5
    end
end
```

In [85]:
```
function nominal_SVM(data,label)
    SVMn = Model(solver=GurobiSolver(OutputFlag=0))
    n = length(label)
    d = size(data,2)
    @variable(SVMn, z[i=1:n]>=0)
    @variable(SVMn, w[j=1:d])
    @variable(SVMn, b)
    for i = 1:n
        @constraint(SVMn, z[i] >= 1 - label[i] * (sum(data[i,j]*w[j] for j=1:d) - b) )
    end
    @objective(SVMn, Min, sum(cost_vector[i]*z[i] for i=1:n) )
    solve(SVMn)
    return [getvalue(w);getvalue(b)]
end
```

```
WARNING: Method definition nominal_SVM(Any, Any) in module Main at In[63]:2
```

Out[85]:
```
nominal_SVM (generic function with 1 method)
```

```
overwritten at In[85]:2.
```

```
In [95]: function RobustSVM_feature(data,label,Γ)
             RSVMf = Model(solver=GurobiSolver(OutputFlag=0))
             n = length(label)
             d = size(data,2)
             @variable(RSVMf, z[i=1:n]>=0)
             @variable(RSVMf, w[j=1:d])
             @variable(RSVMf, b)
             @variable(RSVMf, wmax)
             for j = 1:d
                 @constraint(RSVMf, wmax>=w[j])
             end
             for i = 1:n
                 @constraint(RSVMf, z[i] >= 1 - label[i] * (sum(data[i,j]*w[j] for j=1:d) - b) + Γ*wmax
             end
             @objective(RSVMf, Min, sum(cost_vector[i]*z[i] for i=1:n) )
             solve(RSVMf)
             return [getvalue(w);getvalue(b)]
         end
```

```
Out[95]: RobustSVM_feature (generic function with 1 method)
```

```
In [148]: function RobustSVM_label(data,label,Γ)
              RSVMl = Model(solver=GurobiSolver(OutputFlag=0))
              n = length(label)
              d = size(data,2)
              M = 1e4
              @variable(RSVMl, w[j=1:d])
              @variable(RSVMl, b)
              @variable(RSVMl, q>=0)
              @variable(RSVMl, r[i=1:n]>=0)
              @variable(RSVMl, ϕ[i=1:n]>=0)
              @variable(RSVMl, ξ[i=1:n]>=0)
              @variable(RSVMl, s[1:n],Bin)
              @variable(RSVMl, t[1:n],Bin)


              for i = 1:n
                  @constraint(RSVMl, q + r[i] >= cost_vector[i]*(ϕ[i] - ξ[i]) )
                  @constraint(RSVMl, ξ[i] >= 1 - label[i] * (sum(data[i,j]*w[j] for j=1:d) - b) )
                  @constraint(RSVMl, ξ[i] <= 1 - label[i] * (sum(data[i,j]*w[j] for j=1:d) - b) + M*(1-s
                  @constraint(RSVMl, ξ[i] <= M*s[i])
                  @constraint(RSVMl, ϕ[i] >= 1 + label[i] * (sum(data[i,j]*w[j] for j=1:d) - b) )
                  @constraint(RSVMl, ϕ[i] <= 1 + label[i] * (sum(data[i,j]*w[j] for j=1:d) - b) + M*(1-
                  @constraint(RSVMl, ϕ[i] <= M*t[i])
              end
                  @objective(RSVMl, Min, Γ*q+ sum(cost_vector[i]*ξ[i]+r[i] for i=1:n) )
              solve(RSVMl)
              return [getvalue(w);getvalue(b)]
          end
```

```
WARNING: Method definition RobustSVM_label(Any, Any,
```

```
Out[148]: RobustSVM_label (generic function with 1 method)
```

```
Any) in module Main at In[138]:2 overwritten at In[148]:2.
```

In [149]:
```
function predict(w,data,label)
    n = length(label)
    d = size(data,2)
    predict_label = sign(data*vec(w[1:end-1]) - w[end])
    return sum(abs(predict_label-label))/2/n
end
```

```
WARNING: Method definition predict(Any, Any, Any)
```

Out[149]:
```
predict (generic function with 2 methods)
```

```
in module Main at In[139]:2 overwritten at In[149]:2.
```

In [176]:
```
en = predict(nominal_SVM(X_train, y_train),X_test,y_test);
ef = []
el = []
Γlist = [1,5,10,15,20,25,50,100,120]
for Γ in Γlist
    push!(ef, predict(RobustSVM_feature(X_train, y_train, Γ),X_test,y_test) );
    push!(el, predict(RobustSVM_label(X_train,y_train,Γ),X_test,y_test) );
end
```

In [ ]:
```
plot(Γlist,[en;en;en;en;en;en;en;en;en],label="nominal")
plot(Γlist,ef,label="feature")
plot(Γlist,el,label="label")
legend(loc="upper left")
xlabel("Gamma")
ylabel("Error Rate")
```

In [ ]:

In [ ]:

In [ ]: