Practical session: Optimization and machine learning

Stéphane Canu
scanu@insa-rouen.fr, asi.insa-rouen.fr\~scanu

June 2015, Ecole d'été BasMatI, Porquerolles

Practical session description

This practical session aims at showing how to optimize a non linear cost function under simple constraint. The mixture data proposed in the book "The Elements of Statistical Learning" will be used. It is a two classes simulated dataset. Data in each class were generated from a mixture of 10 lowvariance Gaussian distributions, with individual means themselves distributed as Gaussian.

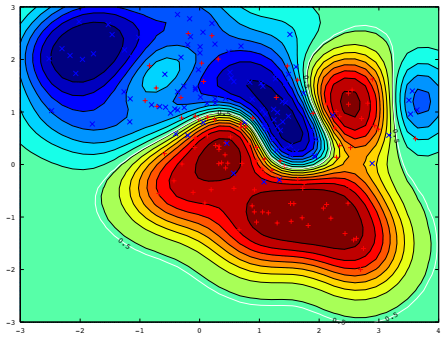

Figure 1: Result of practical session: a sparse kernelized logistic regression.

Ex. 1 — Sparse kernelized logistic regression

1. Load the data from the `TP_data` file. The components of this dataset are:

   $Xapp$ 200 × 2 matrix of training predictors

   $yapp$ 200 × 1 associated class variable; logical vector of TRUES and FALSES - 100 of each

   Xt matrix 6831 x 2 of lattice points x_t in predictor space

   $prob$ vector of 6831 probabilities $\mathbb{P}(C_1|x_t)$ of class TRUE at each lattice point x_t

   $marg$ marginal probability $\mathbb{P}(x)$ at each lattice point

   Use the following code to visualize it.

```
load TP_data
plot(Xapp(yapp==1,1),Xapp(yapp==1,2),'+r'); hold on
plot(Xapp(yapp==0,1),Xapp(yapp==0,2),'xb');
```

   a) Compute the empirical Bayes error rate:

$$\sum_{x_t \in C_2} \mathbb{P}(C_1, x_t) + \sum_{x_t \in C_1} \mathbb{P}(C_2, x_t)$$

   given the Bayes decision rule, that is decide $x_t \in C_1$ when $\mathbb{P}(C_1|x_t) \geq \frac{1}{2}$

```
bayes_error = sum(marg.*(prob.*(prob<0.5)+(1-prob).*(prob>=.5)));
```

2. We propose to write a Matlab function to find the weight w of the linear logistic regression given the data $X \in \mathbb{R}^{n \times q+1}$ and the label $\mathrm{y} \in \{0,1\}^n$, that is the solution of

$$\min_{\mathrm{w} \in \mathbb{R}^{q+1}} J(\mathrm{w}) = \sum_{i=1}^{n} -y_i \left(X_{i\bullet}\mathrm{w}\right) + \log\left(1 + \exp^{X_{i\bullet}\mathrm{w}}\right),$$

where $X_{i\bullet}$ denotes the line i of the matrix X.

a) compute the value of the likelihood function for a given X and w

```
loglik = sum(-yapp.*(X*w) + log(1 + exp(X*w)));
```

b) compute the associated estimated the posterior probabilities $\mathbb{P}(C_1|x)$

```
p = 1./(1+exp(-X*w));
```

c) compute the gradient of the cost function J

```
g = X'*(p - yapp);
```

d) compute the intermediate variable z and update the w.

```
z = W.*(X*w) + (yapp-p); % calcul de la variable intermdiaire
w = (X'*diag(W)*X)\(X'*z); % regression sur z
```

e) compute the vector of the weight of the Hessian

```
W = (p.*(1-p)+sqrt(eps));
```

f)
```
N = 200; % number maximum of iterations
loglik_old = 1/eps; % initialization of the cost
X = [Xapp ones(n,1)];
[n,q] = size(X);
w = zeros(q,1); % initialization of the weight
i = 0; % count the number of iterations
loglik = loglik_old-1; % initialization of the cost
while and((i<N) , (loglik_old - loglik > eps^(1/4)))

    TO BE COMPLETED BY THE PREVIOUS CODE TO UPDATE w

    i = i+1; % count the iterations
    loglik_old = loglik; % to monitor the progress
end
```

g) Note that preconditioning the linear system improve the stability of the solution

```
lambda = sqrt(eps);
I = eye(q);
w = (X'*diag(W)*X + lambda*I)\(X'*z); % add lambda on the diagonal
```

h) write a function called `my_reglog` integrating all above

```
function [w] = my_reglog (X, yapp, N, lambda)
```

3. We propose to write a Matlab function to find the weight w of the kerneled linear logistic regression, that is

$$\min_{\mathrm{w} \in \mathbb{R}^{n+1}} J(\mathrm{w}) = \sum_{i=1}^{n} -y_i f(\mathrm{x}_i) + \log\left(1 + \exp^{f(\mathrm{x}_i)}\right),$$

with

$$f(\mathrm{x}_i) = \sum_{j=1}^{n} w_j K(\mathrm{x}_i, \mathrm{x}_j) + w_{n+1}$$

a) compute the kernel values of the training data (the gram matrix) using the kernel function, and visualize it.

```
kerneloption = .5;
Kapp = kernel (Xapp, Xapp, kerneloption);
figure(2)
imagesc(Kapp)
```

b) use your function `my_reglog` to solve the problem

```
X = [Kapp ones(n,1)];
N = 200;
lambda = 1;
w = my_reglog(X,yapp,N, lambda);
```

c) compute the kernel and the estimated output on the test data. Use it to estimate the error probability.

```
[Kt] = kernel (Xt, Xapp, kerneloption);
Xt = [Kt ones(nt,1)];
ypred = (1./(1+exp(-Xt*w)));

ind1 = find(ypred > 0.5);
probT = 1-prob;
probT(ind1) = prob(ind1);
perrRL = sum(marg.*probT);
```

d) do the nice plot to visualize your solution

```
[xtest1 xtest2] = meshgrid([-1:.05:1.2]*3.5+.5,[-1:0.05:1]*3);
[nn mm] = size(xtest1);
Xtest = [reshape(xtest1 ,nn*mm,1) reshape(xtest2 ,nn*mm,1)];

nt = length(Xtest);
[Kt] = kernel (Xtest, Xapp, kerneloption);
Xt = [Kt ones(nt,1)];
ypred = (1./(1+exp(-Xt*w)));

figure(1);
contourf(xtest1,xtest2,(ypred),50);shading flat;hold on;
plot(Xapp(yapp==1,1),Xapp(yapp==1,2),'+r'); hold on
plot(Xapp(yapp==0,1),Xapp(yapp==0,2),'xb');
[cc, hh]=contour(xtest1,xtest2,ypred,[-100000 0.5],'w','LineWidth',2);
clabel(cc,hh);
axis([-3. 4 -3 3])
```

4. We propose to write a Matlab function to find the weight w of the sparse linear logistic regression, that is the solution of

$$\min_{\mathrm{w}\in\mathbf{R}^{q+1}} J(\mathrm{w}) = \sum_{i=1}^{n} -y_i \left(X_{i\bullet}\mathrm{w}\right) + \log\left(1+\exp^{X_{i\bullet}\mathrm{w}}\right) + \lambda\|\mathrm{w}\|_1.$$

a) compute the proximal gradient sequence

$$\begin{aligned} \mathrm{w}^{k+1} &= \mathbf{prox}_{\rho\|\mathrm{w}\|_1}(\mathrm{w}^k - \rho\nabla g(\mathrm{w}^k)) \\ &= sign(\mathrm{w}^{(g)})\max(0, |\mathrm{w}^{(g)}| - \rho\lambda), \end{aligned}$$

with

$$\mathrm{w}^{(g)} = \mathrm{w}^k - \rho\nabla g(\mathrm{w}^k).$$

```
p = 1./(1+exp(-X*w)); % calcul des probabilites
g = X'*(p - yapp); % calcul du gradient
w = w - pas * g; % intermediate value
w = sign(w).*max(0,abs(w) - pas*lambda); % operateur proximal
```

b) write a function called `my_reglog` integrating the above code in the loop

```
function [w] = my_reglog_l1 (X, yapp, pas, lambda, N)
```

c) use `my_reglog_l1` to estimate the sparse weight of the following double kerneled model

$$f(\mathrm{x}) = \sum_{i=1}^{n} \mathrm{w}_i K_1(\mathrm{x}, \mathrm{x}_i) + \sum_{i=1}^{n} \mathrm{w}_{n+i} K_2(\mathrm{x}, \mathrm{x}_i) + \mathrm{w}_{2n+1}$$

```
kerneloption = .5;
[Kapp1] = kernel (Xapp, Xapp, kerneloption);
[Kapp2] = kernel (Xapp, Xapp, kerneloption/2);
X = [Kapp1 Kapp2 ones(n,1)];
pas = 5*1/svds(X,1)^2;
lambda = 1;
N = 10000;
w = my_reglog_l1 (X, yapp, pas, lambda, N)
```

d) how many component of w are non zero?

```
sizew = length(find(abs(w)>sqrt(eps)));
```

e) what is the error probability estimate on the test set?

```
[Kt1] = kernel (Xt, Xapp, kerneloption);
[Kt2] = kernel (Xt, Xapp, kerneloption/2);
Xt = [Kt1 Kt2 ones(nt,1)];
ypred = (1./(1+exp(-Xt*w)));

ind1 = find(ypred > 0.5);
probT = 1-prob;
probT(ind1) = prob(ind1);
perrRL = sum(marg.*probT);
```

f) do again the nice plot, the same as before.