Untitled5

April 30, 2015

In [ ]:
```
from scipy.optimize import minimize

##This function is the one that will be minimized with respect to w, the model weights
def fun(w,probs,y_true):
    sum = 0
    for i in range(len(probs)):
        sum+= probs[i]*w[i]
    return logloss_mc(y_true,sum)

## First save your model probabilities like so:
probs=[p1,p2,p3,p4]

## w0 is the initial guess for the minimum of function 'fun'
## This initial guess is that all weights are equal
w0 =np.ones(len(probs))/(len(probs))

## This sets the bounds on the weights, between 0 and 1
bnds = tuple((0,1) for w in w0)

## This sets the constraints on the weights, they must sum to 1
## Or, in other words, 1 - sum(w) = 0
cons = ({'type':'eq','fun':lambda w: 1-sum(w)})

## Calls the minimize function from scipy.optimize
## ------------------------------------------------
## fun is the function defined above
## w0 is the initial estimate of weights
## (probs, y_true) are the additional arguments passed to 'fun'; probs are the probabilities,
##                 y_true is the expected output for your training set
## method = 'SLSQP' is a least squares method for minimizing the function;
##   There are other methods available, but I don't know enough about the theory to make recomm
## ------------------------------------------------
weights = minimize(fun,x0,(probs,y_true),method='SLSQP',bounds=bnds,constraints=cons).w

## As a sanity check, make sure the weights do in fact sum to 1
print("Weights sum to %0.4f:" % weights.sum())

## Print out the weights
print(weights)

## This will combine the model probabilities using the optimized weights
y_prob = 0
for i in range(len(probs)):
```

```
y_prob += probs[i]*weights[i]
```