Error exponents for compound BSC

Aditya Mahajan

February 17, 2010

This article contains R source code used in the following paper

> Aditya Mahajan and Sekhar Tatikonda, "Opportunistic capacity and error exponent regions for compound channels with feedback", submitted to IEEE Transactions on Information Theory, 2010

This code is in written in R using the Sweave format. To process the file run

```
Sweave("bsc-code.Rnw")
```

We consider a compound channel consisting of two BSCs with complementary crossover probabilities, p and $(1-p)$, where $0 < p < 1/2$ and p is known to the transmitter and the receiver. We denote this compound channel by

$$\mathcal{Q}_p \coloneqq \{\mathrm{BSC}_p, \mathrm{BSC}_{1-p}\}$$

where BSC_p denotes a binary symmetric channel with crossover probability p. For convenience, we will index all variables by p and $(1-p)$ rather than by 1 and 2. For binary symmetric channel, the capacity and B_Q term of Burnashev exponent are given by

$$C_p = C_{1-p} = 1 - h(p)$$

```
bsc.C <- function(p)
{
  return (1 - binary.h(p))
}
```

and

$$B_p = B_{1-p} = D(p\|1-p)$$

```
bsc.B <- function(p)
{
  return (binary.D(p,1-p))
}
```

where $h(p) = -p\log p - (1-p)\log(1-p)$ is the binary entropy function

```
binary.h <- function(p)
{
  return ( -p*log2(p) - (1-p)*log2(1-p))
}
```

and $D(p\|q) = -p\log(p/q) - (1-p)\log((1-p)/(1-q))$ is the binary Kullback-Leibler function.

```
binary.D <- function(x,y)
{
  return ( x*log2(x/y) + (1-x)*log2((1-x)/(1-y)))
}
```

We choose the all zero sequence as a training sequence and estimate the channel based on the type of the output sequence. If the empirical frequency of ones in the output is less than q, $p < q < 1-p$, the channel is estimated as BSC_p; otherwise the channel is estimated as BSC_{1-p}. For this class of channel estimation rules, the estimation error probability is bounded by the tail of the probability of the sum of independent random variables. From Hoeffding's inequality, the exponents of the estimation errors are given by

$$T_p = D(q\|p), \quad T_{1-p} = D(q\|1-p).$$

Suppose we want to communicate at rate (R_p, R_{1-p}), $R_p < C_p$ and $R_{1-p} < C_{1-p}$, using the coding scheme of the paper. Let q_m and q_c be the estimation thresholds for the message and control mode. The lower bound of Proposition 2 simplifies to

$$E_p \geq \frac{D(q_c\|p)D(p\|1-p)}{D(q_c\|p) + D(p\|1-p)}(1-\gamma_p)$$

and

$$E_{1-p} \geq \frac{D(q_c\|1-p)D(p\|1-p)}{D(q_c\|1-p) + D(p\|1-p)}(1-\gamma_{1-p})$$

where $\gamma_p = R_p/C_p$ and $\gamma_{1-p} = R_{1-p}/C_{1-p}$.

Now, we want to choose q_c such that $E_p = E_{1-p}$ which is equivalent to choosing q_c such

$$\varphi(q_c, p) = \frac{(1-\gamma_p)}{(1-\gamma_{1-p})}$$

where

$$\varphi(q,p) = \frac{1 + D(p\|1-p)/D(q\|p)}{1 + D(p\|1-p)/D(q\|1-p)}$$

```
compoundBsc.phi <- function(p,q)
{
  num = 1 + binary.D(p, 1-p) / binary.D(q, p)
  den = 1 + binary.D(p, 1-p) / binary.D(q, 1-p)
  return (num/den)
}
```

This means that $q_c = 0.5$, which maximally distinguishes between BSC_p and BSC_{1-p} is optimal only when $\gamma_p = \gamma_{1-p}$. For other values of γ_p and γ_{1-p}, we need to invert $\varphi(q_c, p)$ to determine the value of q_c.

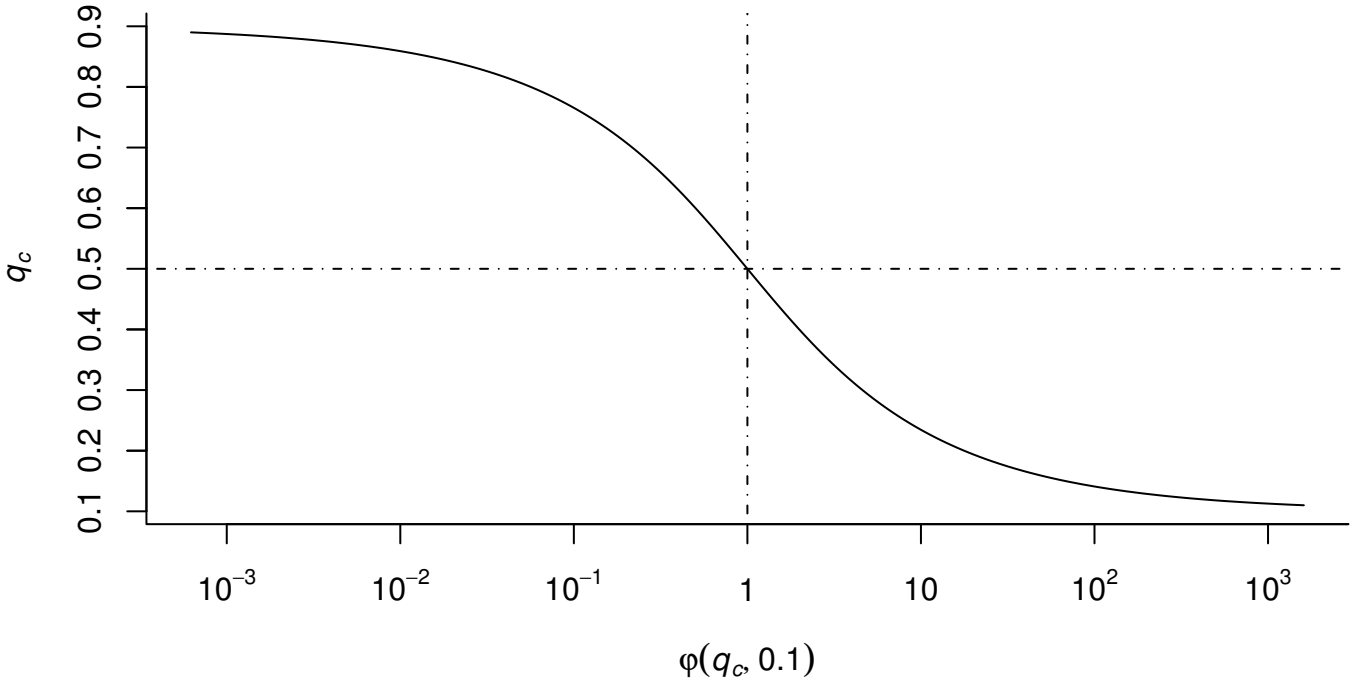
```
compoundBsc.E <- function(p, γ_1, γ_2) {
  # if (0 > γ_1 || 0 > γ_2 || 1 < γ_1 || 1 < γ_2)
  #   stop("γ out of bound")

  exponent <- function (p,q,γ)
  {
    num = binary.D (q,p) * binary.D (p, 1-p) * (1 - γ)
    den = binary.D (q,p) + binary.D (p, 1-p)
```

```
    return (num/den)
  }
  findQ <- function (q)
  {
    return (exponent(p,q,γ_1) - exponent(1-p,q,γ_2))
  }
  eps = 10e-3
  q = uniroot(findQ, upper = 1-p - eps, lower = p + eps)$root
  # if (abs (exponent(p,q,γ_1) - exponent(1-p,q,γ_2)) > eps )
  #   warning(sprintf("q not within %f accuracy for p=%f, γ_1=%f, γ_2=%f",
  #                   eps, p, γ_1, γ_2 ))
  return (list(exp=exponent(p,q,γ_1), q=q))
}
```

The code below gives the plot of $\varphi(q_c, 0.1)$ for different values of q_c.

```
# The affect of varying threshold q
compoundBsc.plotPhi <- function (p=0.1)
{
  eps = 10e-3
  q   = seq(p+eps, 1-p-eps, by=0.001)
  lab = parse(text=sprintf("varphi(italic(q[c]), %.1f)", p))

  pdf ("threshold.pdf", width = 7.5, height = 4.5)  # Output file
  par (lty = "solid", lwd = 1, bty = "l")           # An "L" box
  plot(compoundBsc.phi(p, q), q,
       lty  = "solid",
       lwd  = 1      ,
       type = "l"    ,
       log  = "x"    ,
       xaxt = "n"    ,
       ylab = expression(italic(q[c])) ,
       xlab = lab    )

  abline(v = 1,  lty = "1343")
  abline(h = 0.5,lty = "1343")

  # The axis are fine tuned to p=0.1
  axis(2,at=seq(p,1-p, by = 0.1))

  axis(1,at=c(1e-3,1e-2,1e-1,1,1e1,1e2,1e3,1e4),
  labels=c(
             expression(10^-3),
             expression(10^-2),
             expression(10^-1),
             "1"              ,
             "10"             ,
             expression(10^2) ,
             expression(10^3) ,
```

```
            expression(10^4)))
  dev.off()
  embedFonts("threshold.pdf", options="-dPDFSETTINGS=/prepress")
}
```



plot

For different values of γ_p and γ_{1-p}, the optimal value of q is given by the function below.

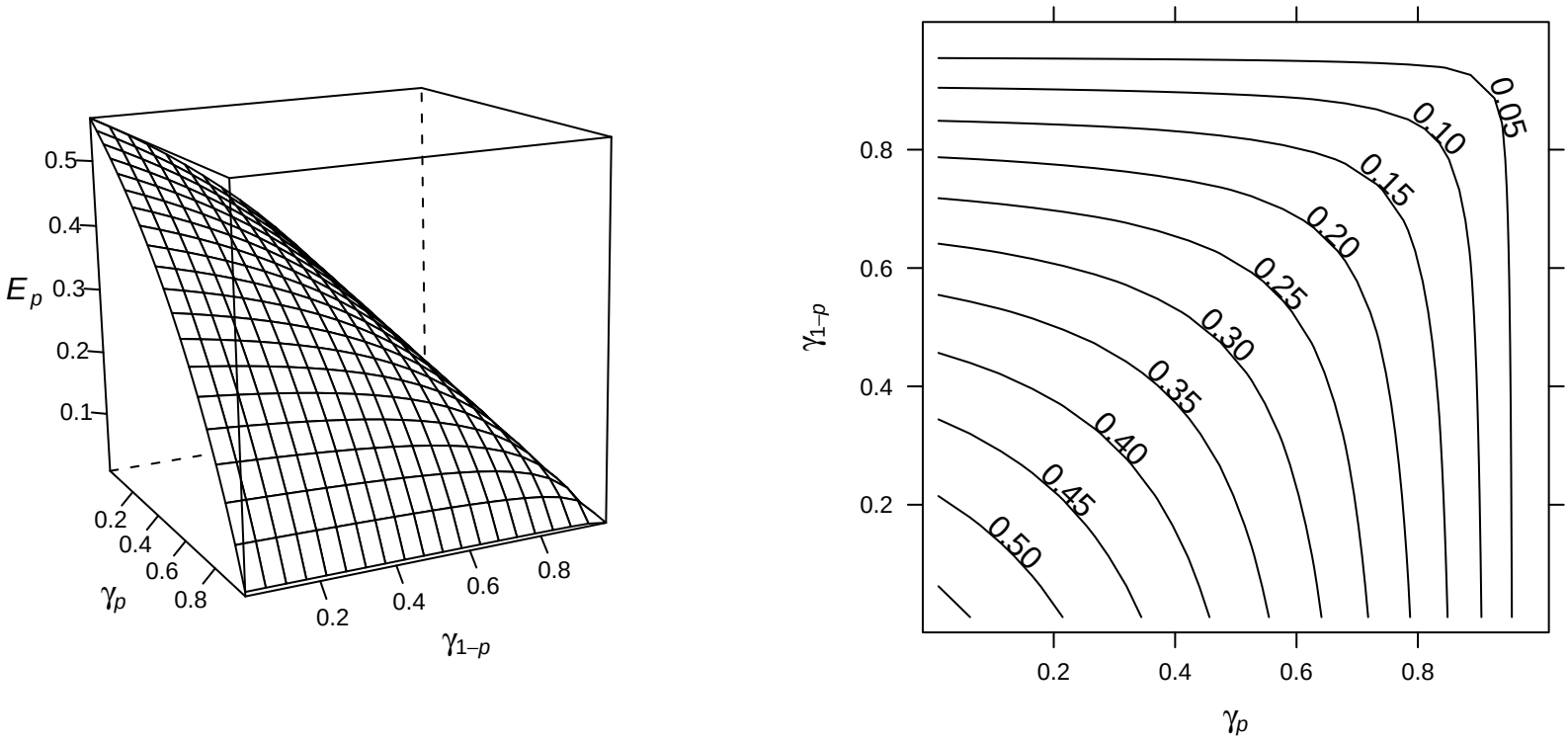
```
# Show the table of q and E values
compoundBsc.showValues <- function(p = 0.1,
                                   γ_1 = rep(0.5,9),
                                   γ_2 = seq(0.1, 0.9, length=9) )
{
  len    = min(length(γ_1), length(γ_2))
  output = data.frame(γ_1 = γ_1,γ_2 = γ_2,  q = rep(NA, len), exp = rep(NA, len))
  for (i in 1:len)
  {
    values        = compoundBsc.E(p, γ_1[i], γ_2[i])
    output$q[i]   = values$q
    output$exp[i] = values$exp
  }
  return (format(output, width=8, digits=4))
}
```

```
      γ_1      γ_2        q      exp
1     0.5      0.1   0.5861  0.36663
2     0.5      0.2   0.5695  0.35112
3     0.5      0.3   0.5501  0.33295
4     0.5      0.4   0.5273  0.31142
5     0.5      0.5   0.5000  0.28551
6     0.5      0.6   0.4666  0.25373
7     0.5      0.7   0.4247  0.21391
8     0.5      0.8   0.3698  0.16284
9     0.5      0.9   0.2918  0.09526
```

The next function gives the error exponent for different values of γ_p and γ_{1-p}.

```
# Error exponents for different values of γ
compoundBsc.plotExp <- function(p=0.1)
{
  library (lattice)
  p = 0.1
  eps = 10e-3
  γ_1 = seq(eps,1-eps,length=20)
  γ_2 = seq(eps,1-eps,length=20)

  exponent <- function(g1, g2)
  {
    return(compoundBsc.showValues(p,g1,g2)$exp)
  }

  z = outer(γ_1, γ_2, exponent)

  pdf ("exponent.pdf", width = 9, height = 4.5)  # Output file

  points = expand.grid(x=γ_1, y=γ_2)
  points$z = as.numeric(exponent(points$x, points$y))

  # Remove the box around the wireframe
  trellis.par.set("axis.line", list(col = "transparent"))
  plot1 <- wireframe(z ~ x * y, data=points,
            screen = list (z = -60, x = -75, y=-5),
            xlab = expression(gamma[italic(p)]),
            ylab = expression(gamma[1-italic(p)]),
            zlab = expression(italic(E)[italic(p)]),
            scales = list(cex  = 0.75, col = "black", arrows=FALSE),
            shades = TRUE, colorkey=FALSE,
           )
  plot(plot1, split = c(1, 1, 2, 1))

  trellis.par.set("axis.line", list(col = "black"))
  plot2 <- contourplot(z ~ x * y, data=points,
            screen = list (z = -60, x = -75, y=-5),
```

```
            xlab = expression(gamma[italic(p)]),
            ylab = expression(gamma[1-italic(p)]),
            zlab = expression(italic(E)[italic(p)]),
            cuts = 10, aspect = "fill",
            scales = list(cex  = 0.75, col  = "black", arrows=FALSE),
            region = FALSE, colorkey=FALSE,  pretty=TRUE,
           )

  plot(plot2, split = c(2, 1, 2, 1), newpage = FALSE)
}

null device
          1
```



values

The best possible error exponent for the compound channel $\mathcal{Q}_p$ is not known. Nonetheless, we can compare the error exponent of our scheme with two simple upper bounds (We only evaluate the bounds for BSC_p. The bounds for BSC_{1-p} are symmetric). The first upper bound on the error exponent is given by the error exponent when both transmitter and receiver know that the channel is BSC_p. That, the first upper bound E_p' equals to the Burnashev exponent of BSC_p, $B_p(1-\gamma_p)$.

```
compound.burnashev <- function(p,R)
{
```

```
  γ = R/bsc.C(p)
  return (bsc.B(p) * (1-γ))
}
```

Since decreasing the rate of transmission cannot decrease the error exponent, the second upper bound on the error exponent is given by the error exponent of communicating at zero-rate. Zero-rate communication over unknown DMCs with feedback was considered by (Tchamkerten Telatar, 2005), which provided an upper bound on the error exponent. For $\mathcal{Q}_p$ this upper bound evaluates to $E_p'' = \frac{1}{2}B_p$.

Combine these two upper bounds to obtain a unified upper bound

$$E^*(\gamma) = B_p(1 - \max(\tfrac{1}{2}, \gamma_p)).$$

```
compound.upper <- function(p,R)
{
  return (pmin(0.5*bsc.B(p), compound.burnashev(p,R)) )
}
```

The error exponent of our scheme for $\gamma_p = \gamma_{1-p}$ is given by

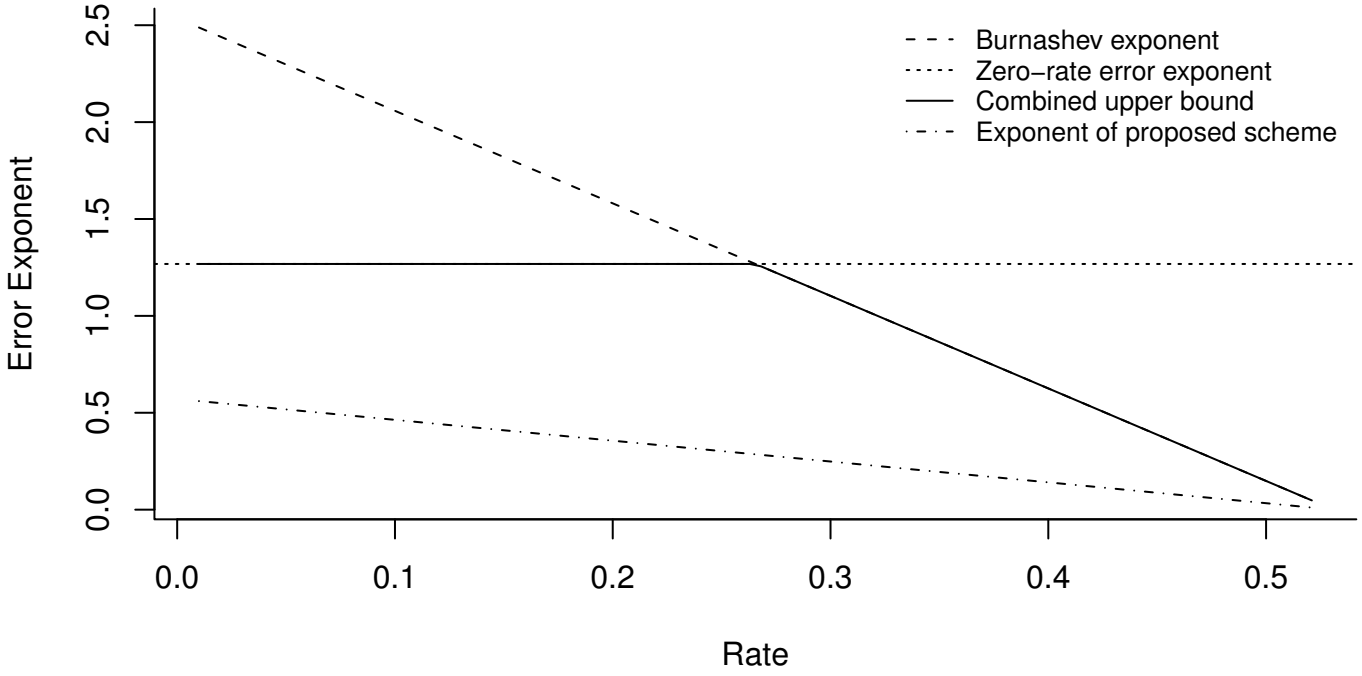
$$E_p \geq \frac{D(0.5\|p)D(p\|1-p)}{D(0.5\|p) + D(p\|1-p)}(1-\gamma_p)$$

```
compound.training <- function(p,R)
{
  γ = R/bsc.C(p)
  q = 0.5
  a = binary.D(0.5, p)
  b = binary.D(p, 1-p)
  return (a*b*(1-γ)/(a+b))
}
```

These upper bounds, along with the error exponent of our scheme for $\gamma_p = \gamma_{1-p}$, are plotted using the function below.

```
# Upper bound on error exponent
compoundBsc.plotBound <- function (p=0.1)
{
  eps = 10e-3
  R <- seq(eps, bsc.C(p) - eps, length=100)

  pdf("upper.pdf", width=7.5, height=4.5)
  par (lty = "solid", lwd = 1, bty = "l")           # An "L" box
  plot(R, compound.burnashev(p,R),
       type = "l"  ,
       lty  = "44" ,
       lwd  = 1    ,
       xlab = "Rate" ,
       ylab = "Error Exponent")

  abline (h = 0.5*bsc.B(p), lty = "22")
```

```
  lines (R, compound.upper(p,R), lty = "solid")

  lines (R, compound.training(p,R), lty="1343")
  legend("topright",
         c("Burnashev exponent"          ,
           "Zero-rate error exponent"    ,
           "Combined upper bound"        ,
           "Exponent of proposed scheme "),
        lty = c("44", "22", "solid", "1343"),
        cex = 0.8, lwd=1, bty="n")

  dev.off()
  embedFonts("upper.pdf", options="-dPDFSETTINGS=/prepress")
}
```



exponents