James-Stein Simulation

Maximum Likelihood Estimation (MLE) is a technique that lies implicit in most commonly used statistical models, including Oridinary Least Squares regression. Specificlly, if we have a joint probabilty distribution of $\{x_1, x_2, \dots\}$ with a true parameter $\theta = \{\theta_1, \theta_2, \dots\}$ and we observe the sample $\mathbf{x}$ we have the MLE estimate:

$$\hat{\theta} = \arg\max_{\theta} \widehat{L}_n(\theta \,|\mathbf{x})$$

This is very intuitive, as we are simply selecting the parameter estimates that are most likely to have generated the sample that we observed. For instance, the below function generates a multivariate normal distribution with a specified number of dimensions, all with mean zero and a covariance matrix that is a scalar multiple of the identity matrix (specifying uncorrelated variables).

```
theta.sim = function(var.scale, dim){
  rmnorm(n = 1, mean = rep(0, dim), varcov = var.scale*diag(dim))
}
```

So if I call the function:

```
set.seed(1)
theta.sim(var.scale = 2, dim = 3)
```

```
## [1] -0.8859395  0.2597109 -1.1817573
```

I am given a single sample from the distribution:

$$\mathbf{X} \;\sim\; \mathcal{N}(\boldsymbol{\theta},\, \boldsymbol{\Sigma})$$

where we have:

$$\boldsymbol{\theta} = (0, 0, 0)$$

and

$$\boldsymbol{\Sigma} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

So if I observed this sample, what would MLE tell me to select as my estimate of $\boldsymbol{\theta}$? Since I know that the most likely value to be observed is the mean, my estimate would simply be be my observation itself:

$$\hat{\boldsymbol{\theta}}_{\mathrm{MLE}} = (-0.8859395, 0.2597109, -1.1817573)$$

A natural way to quantify the success of our estimation is by the squared difference between our estimate and the true parameters. This is ususally refered to as risk or squared loss. In our case:

$$\mathrm{Risk} = \frac{1}{p}\sum_{i=1}^{p}(x_i - \theta_i)^2$$

where p is the number of parameters considered and $\mathbf{x} = \{x_1, x_2, \dots\}$ is our observed sample.

MLE performs rather well in terms of this measurement. In fact, we can show that for a scaled normal distribution $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\theta},\, I_n)$ that the expected value of this risk is equal to 1, that is:

$$\mathrm{E}\left[\frac{1}{p}\sum_{i=1}^{p}(x_i - \theta_i)^2\right] = \frac{1}{p}\sum_{i=1}^{p}\mathrm{E}\left[(x_i - \theta_i)^2\right] = 1$$

This also makes a sort of intuitive sense, in that if we select a single observation as our estimate of a multivariate normal distribution that we would expect the total deviation to remain within roughly one standard deviation of the mean.

Surprisingly in 1961, the statistician Charles Stein and his student Willard James produced a different sort of estimation method that can be shown to have uniformly lower risk than MLE for three or more dimensions. What takes this from surprising to nearly unbelievable is that this method works by taking the original MLE estimate and shrinking it by a factor that is inversely proportional to the squared norm of our observation. Think about how unintuitve this is for a moment. Even under the supposition that our variables are uncorrelated, the James-Stein estimator uses the entire sample to determine how much to shrink our estimate.

Still considering the above case of taking a single sample from the normal distribution $\mathcal{N}(\boldsymbol{\theta},\, I_n)$ the James-Stein estimator is the following:

$$\hat{\boldsymbol{\theta}}_{\mathrm{JS}} = \left(1 - \frac{p-2}{\|\mathbf{x}\|^2}\right)\mathbf{x}$$

For instance, above we had $\hat{\boldsymbol{\theta}}_{\mathrm{MLE}} = (-0.8859395, 0.2597109, -1.1817573)$. Our shrinkage factor would be:

$$\left(1 - \frac{1}{-0.8859395^2 + 0.2597109^2 + -1.1817573^2}\right) = .555336$$

giving:

$${\hat{\boldsymbol{\theta}}_{\mathrm{JS}}}^{T} = .555336 \begin{bmatrix} -0.8859395 \\ 0.2597109 \\ -1.1817573 \end{bmatrix} = \begin{bmatrix} -0.4919941 \\ 0.1442268 \\ -0.6562724 \end{bmatrix}$$

(I have written the transpose only as a space consideration)

The below function repeats this same process for multiple samples. It intakes the true parameter of $\boldsymbol{\theta}$, takes the specified number of samples from the corresponding multivariate normal distribution (still with the identity for covariance) and returns the MLE and James-Stein estimates for each observation. The argument return.risk or return.estimates specifies if I would like to view the results or just return the average risk.

```r
risk = function(theta.vec, n.sample, return.risk = FALSE, return.estimates = FALSE){

  risk.mat = matrix(ncol = 2, nrow = n.sample)
  p = length(theta.vec)

  estimate.mat = matrix(ncol = p*2, nrow = n.sample)

  for (trial in 1:n.sample){
    mle = rmnorm(n = 1, mean = theta.vec, varcov = diag(p))
    risk.mle = sum((mle - theta.vec)^2)/p

    JS = (1 - (p-2)/sum(mle^2))*mle
    risk.JS = sum((JS - theta.vec)^2)/p

    risk.mat[trial,] = c(risk.mle, risk.JS)

    if(return.estimates){
      estimate.mat[trial,] = c(mle, JS)
    }

  }

  if (return.risk){
    risk.temp.df = as.data.frame(risk.mat)
    colnames(risk.temp.df) = c("MLE Estimate Risk", "JS Estimate Risk")

    return(risk.temp.df)

  } else if (return.estimates){

    estimate.temp.df = as.data.frame(estimate.mat)
    colnames(estimate.temp.df) = c(paste("MLE Theta Hat", 1:p),
                                   paste("JS Theta Hat", 1:p))

    return(estimate.temp.df)

  } else {

    return(colMeans(risk.mat))

  }
}
```

For example, I can take a sample of 5 and compare MLE and JS:

```r
set.seed(1)
risk(theta.vec = c(0, 0, 0), n.sample = 5, return.estimates = TRUE)
```

```
##   MLE Theta Hat 1 MLE Theta Hat 2 MLE Theta Hat 3 JS Theta Hat 1
## 1      -0.6264538       0.1836433      -0.8356286    -0.06933085
## 2       1.5952808       0.3295078      -0.8204684     1.11573717
## 3       0.4874291       0.7383247       0.5757814     0.04997266
## 4      -0.3053884       1.5117812       0.3898432    -0.18471596
## 5      -0.6212406      -2.2146999       1.1249309    -0.52648590
##   JS Theta Hat 2 JS Theta Hat 3
## 1     0.02032416    -0.09248062
## 2     0.23045728    -0.57383445
## 3     0.07569522     0.05903080
## 4     0.91440971     0.23579897
## 5    -1.87690293     0.95335090
```

view the risk for each individual sample:

```r
set.seed(1)
risk(theta.vec = c(0, 0, 0), n.sample = 5, return.risk = TRUE)
```

```
##   MLE Estimate Risk JS Estimate Risk
## 1         0.3748148      0.004590835
## 2         1.1088882      0.542421990
## 3         0.3714115      0.003903890
## 4         0.8435740      0.308622084
## 5         2.1854350      1.569609985
```

and see that the average risk of the MLE estimate is close to one, while the James-Stein estimator is significantly lower:

```r
set.seed(1)
risk(theta.vec = c(0, 0, 0), n.sample = 5)
```

```
## [1] 0.9768247 0.4858298
```

In the case of MLE, the expected value of the risk is uniformly one. For the James-Stein estimator however, this expectation is not a uniform value, instead jointly dependant upon the number of variables and how close the mean vector is to zero.

To demonstrate this effect, I can run the below simulation that calculates the average sample risk for different combinations of covariance of the means of our variable and the number of variables under consideration. In other words, I am treating the true parameter $\boldsymbol{\theta}$ as a random variable itself.

Explicity, I first sample:

$$\boldsymbol{\theta} \sim \mathcal{N}(0_n, \sigma I_n)$$

where 0_n is the vector of length n: $(0, 0, \ldots 0)$ and σ is a constant. Our multivariate normal distribution that we will sample from is:

$$\mathbf{X} \sim \mathcal{N}(\boldsymbol{\theta}, I_n)$$

What we will vary are the hyperparameters n and σ, which will effect the number of dimensions considered and the spread of our vector $\boldsymbol{\theta}$. For each combination of dimension and σ, we will sample 5000 points and return the average risk.

```r
variance.vec = seq(from = .01, to = 5, by = .01)
n.var.vec = seq(from = 4, by = 4, length.out = 5)

df = data.frame(matrix(ncol = 4, nrow = 0))
colnames(df) = c("Number.Variables", "Variance.Scale", "risk.MLE", "risk.JS")

for (var.num in n.var.vec){

  for(variance in variance.vec){

    theta.vec = theta.sim(dim = var.num, var.scale = variance)
    risk.loop = risk(theta.vec = theta.vec, n.sample = 5000)

    df[nrow(df) + 1,] = c(var.num, variance, risk.loop)

  }

}
```

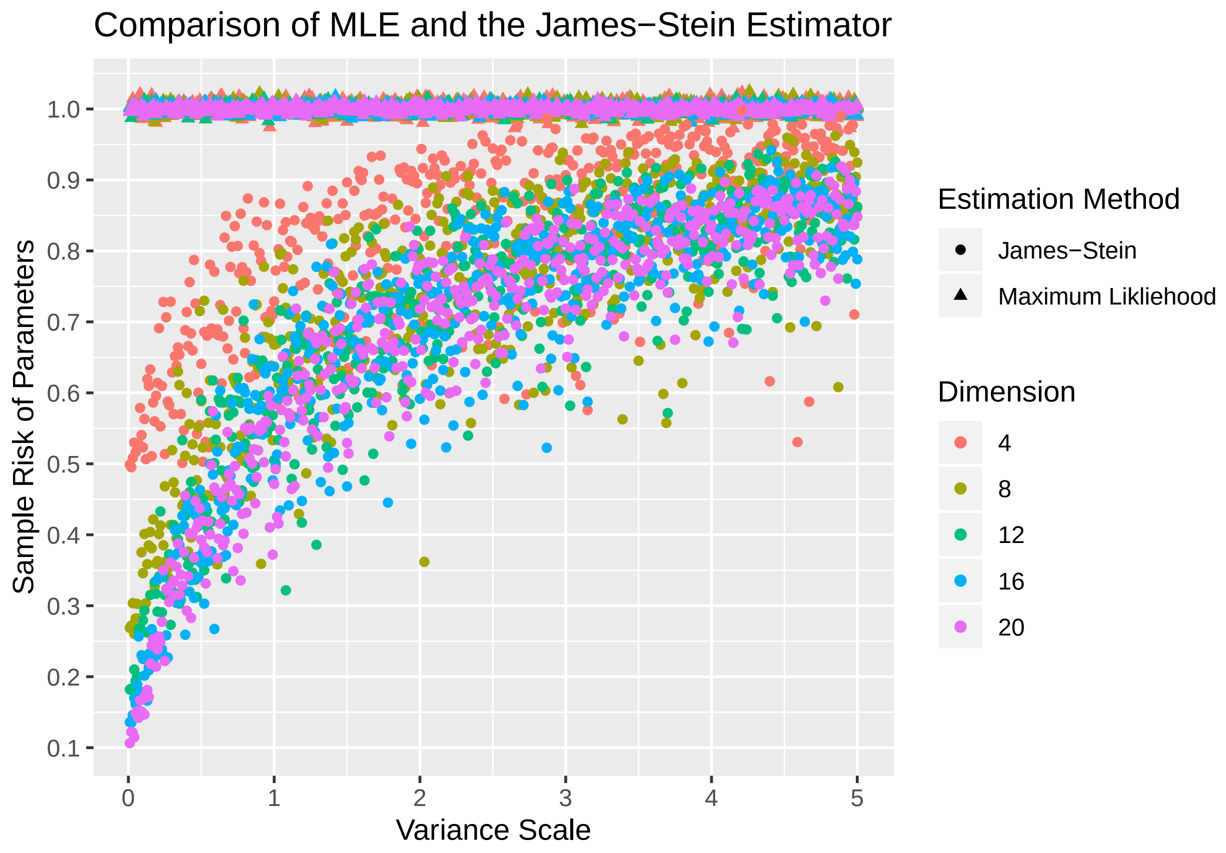
Having run the simulation, we can see that we have sampled 5000 points each from a total of 2500 different normal distributions, for each returning the average risk for the MLE and James-Stein estimators. Looking at the first few rows:

```r
head(df)
```

```
##   Number.Variables Variance.Scale  risk.MLE   risk.JS
## 1                4           0.01 1.0032960 0.4989967
## 2                4           0.02 1.0064331 0.4952396
## 3                4           0.03 1.0143297 0.5082744
## 4                4           0.04 1.0009178 0.5297626
## 5                4           0.05 1.0096028 0.5164215
## 6                4           0.06 0.9954835 0.5267456
```

Graphing our results, we can now compare the James-Stein and MLE Estimates. Here the x-axis is our constant multiple σ that determines the dispersion of $\boldsymbol{\theta}$, the y-axis is the (sample) average risk, and the colors indicate the dimension for each sample. As expected, we see that the risk of the MLE estimations remains roughly constant regardless of the number of dimensions or the scale of our $\boldsymbol{\theta}$.

For the James-Stein Estimator, the result is much different. We see that for exteremely small values of σ, in other words multivariate normal distributions where the means are all very close to zero, the JS estimator has a significant advantage over MLE. Likewise, we can see that as the number of predictors increases the JS estimator also performs better. Across all dimensions, we see that the risk of the JS estimate approaches that of MLE as dispersion of our means increases.



In fact, with a bit of work we can quantify this relationship for the James-Stein estimator as follows:

$$\text{Risk}_{\text{JS}} = 1 - \frac{(p-2)^2}{p}\,\text{E}\left[\frac{1}{\|\mathbf{x}\|^2}\right]$$

where as above $\mathbf{x}$ represents our sample.

Given all of this information about how the James-Stein estimator dominates MLE in terms of risk, why is MLE still used? Besides the history and ease of understanding that comes with MLE, there are other factors to consider. The most pressing is that while we a guaruntee regarding the sum of the squared loss for all parameters, we have no such guarentee regarding each individual parameter. Especially if one dimension is of a very different scale, that parameter could be estimated incorrectly by a wide margin, even though the overall risk of of using James-Stein is lower.

Using the same risk function from earlier, we see that because the last dimension has a mean that is so different from the others that most of the benefit of the James-Stein estimator has been lost. As an example, consider the case where $\boldsymbol{\theta} = (0, 0, 10)$:

```r
set.seed(1)
risk(theta.vec = c(0, 0, 10), n.sample = 5000, return.risk = FALSE)
```

```
## [1] 1.003480 1.001498
```

Why is this the case? Remember that the shrinkage factor is determined by the reciprocal of the squared norm of the vector $\mathbf{x}$, our sample observation. Since one dimension is so different from the others, this shrinkage is no longer working as well as it previously did.

Now, instead of just looking at the aggregated risk, I want to see how far each individual parameter estimate strays from the true value. For the ease of presentation, I will first restrict myself to the case with three dimensions and instead varying the value of the third dimension's mean.

As an example let $\boldsymbol{\theta} = (0, 0, 10)$ and look at the same sample:

```r
set.seed(1)
individual.estimates = risk(theta.vec = c(0, 0, 10), n.sample = 5000, return.estimates = TRUE)
head(individual.estimates)
```

```
##   MLE Theta Hat 1 MLE Theta Hat 2 MLE Theta Hat 3 JS Theta Hat 1
## 1     -0.62645381      0.18364332        9.164371    -0.61903242
## 2      1.59528080      0.32950777        9.179532     1.57692679
## 3      0.48742905      0.73832471       10.575781     0.48310135
## 4     -0.30538839      1.51178117       10.389843    -0.30262037
## 5     -0.62124058     -2.21469989       11.124931    -0.61642681
## 6     -0.04493361     -0.01619026       10.943836    -0.04455844
##   JS Theta Hat 2 JS Theta Hat 3
## 1     0.18146776       9.055804
## 2     0.32571672       9.073919
## 3     0.73176939      10.481883
## 4     1.49807851      10.295671
## 5    -2.19753898      11.038728
## 6    -0.01605508      10.852462
```

Now we can find the difference for each individual estimate from the true $\boldsymbol{\theta}$:

```r
mle.ind.loss = sweep(individual.estimates[,1:3], 2, c(0, 0, 10))^2
js.ind.loss  = sweep(individual.estimates[,4:6], 2, c(0, 0, 10))^2
```

And substact the averages to see how such better/worse each method does:

```r
as.vector(colMeans(js.ind.loss) - colMeans(mle.ind.loss))
```

```
## [1] -0.02006558 -0.01914597  0.03326705
```

We see that for the third dimension which we assigned a much higher mean that the value of the individual mean was better predicted by MLE than by JS (as the sign of our the difference in squared loss between JS and MLE is positive). Realize that the overall expected loss of the estimate is still lower (by a very close margin):

```r
set.seed(1)
risk(theta.vec = c(0, 0, 10), n.sample = 5000)
```

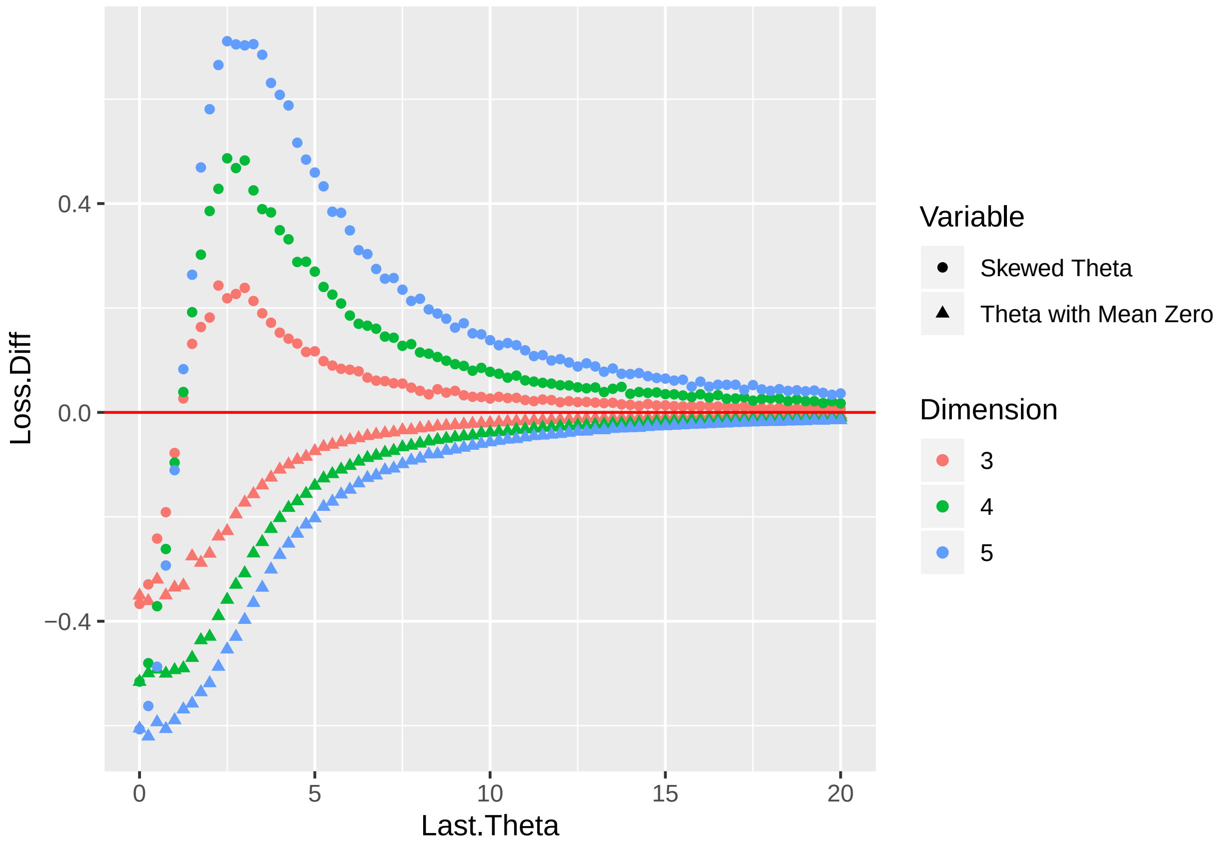
```
## [1] 1.003480 1.001498
```

The takeaway is that while the JS will always have an overall lower expected loss for our parameters, that this is not true of the individual parameters themselves.

We can repeat this for an arbitray number of dimensions, only looking at the varying dimension and one of the dimensions with zero mean for the sake of clarity. Here, I have calculated for each dimension the differnce expected squared loss for each individual parameter between the James-Stein estimator and MLE.

```r
set.seed(1)
third.dim.mean = seq(from = 0, to = 20, by = .25)
loss.df  = as.data.frame(matrix(ncol = 4, nrow = 0))
colnames(loss.df) = c("Dimension", "Last.Theta",
                      "Loss.Diff.Mean.Zero", "Loss.Diff.Last.Theta")

for (dim in 3:5){

  for (mean in third.dim.mean){

    theta.vec = c(rep(0, dim-1), mean)

    individual.estimates = risk(theta.vec = theta.vec,
                                n.sample = 10000,
                                return.estimates = TRUE)

    mle.ind.loss = sweep(individual.estimates[,1:dim], 2, theta.vec)^2
    js.ind.loss  = sweep(individual.estimates[,(dim+1):(2*dim)], 2, theta.vec)^2

    loss.vec = as.vector(colMeans(js.ind.loss) - colMeans(mle.ind.loss))

    loss.last = loss.vec[dim]
    loss.theta.zero = loss.vec[dim-1]

    loss.df[nrow(loss.df) +1,] = c(dim, mean, loss.theta.zero, loss.last)

    }

}
```

Graphing each estimation as a fucntion of the mean of the dimension that differs from zero (so that below the horizintal line at zero JS dominates while above MLE dominates) we see that JS quickly becomes a poor estimate of this parameter. As this dimension grows in scale, it pulls the estimates of the other variables closer to the MLE estimate. On the other hand the estimate of our other variables with mean zero becomes much better as dimensionality increases, consistent with our original experiment.



In fact, this determination of regions where the individual risk of our parameters is better than MLE can be formulated exactly, but requires much more additional mechanics.

The question then becomes, when should we use the James-Stein estimator? In this example we have used a vector of zeros as our baseline to shrink towards, but nothing is special about this particuliar direction of shrinkage. We could instead shrink towards another given vector, usually determined by a sample estimate of our parameters.

What does matter is the effect of the relative scale of our parameters, as a proxy for the noncentrality of our means. What we have seen is that the James-Stein estimator, while always an imporovement in the overall risk of estimating parameters from a multivariate normal distribution, is particuliarly effective when all of our variables are very close in terms of scale and when there is a large number of variables. Additionally, if the parameters themselves are not of particuliar concern, we should certainly use James-Stein estimation. This method applies to not only sampling from multivariate normal distibutions, but may also be used for shrinkage estimations of linear regression.