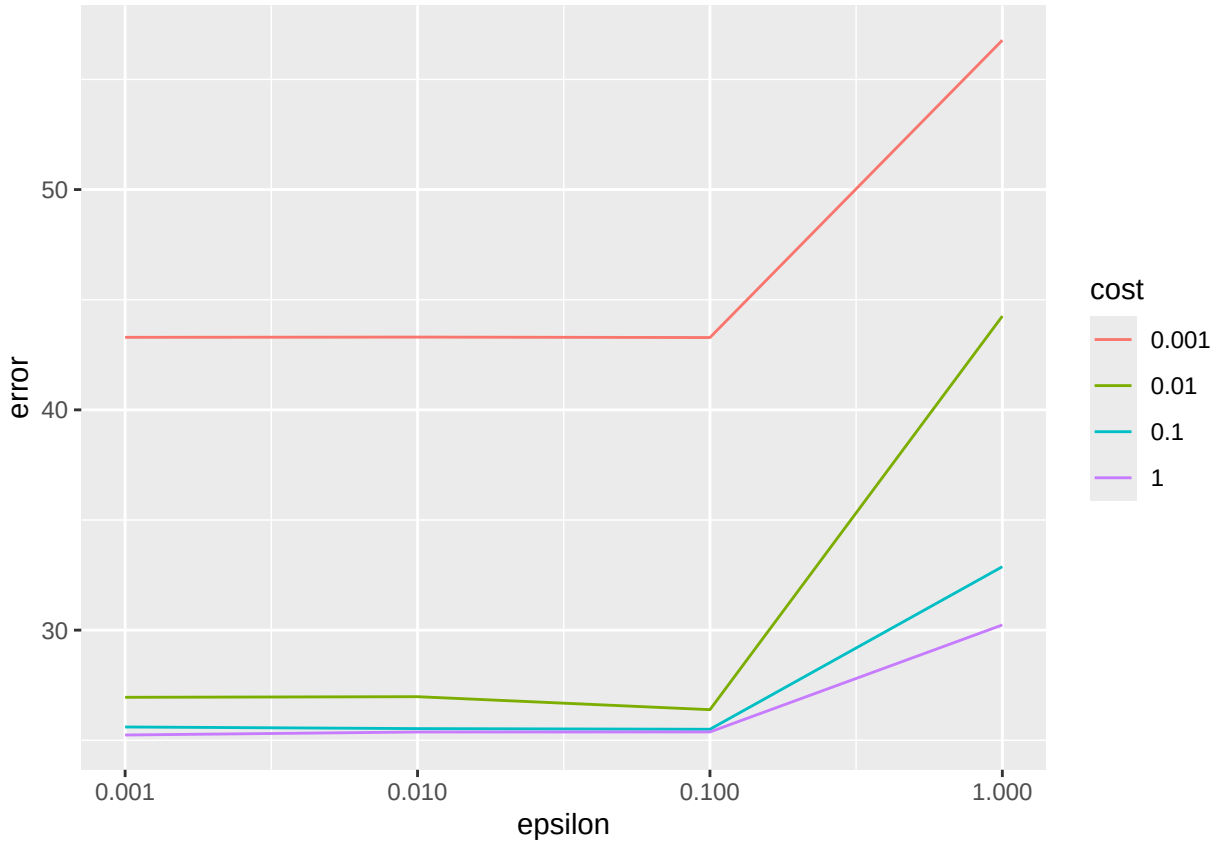
Weekly Assignment 8

Gaspard Gaches (4645251)

2025-04-22

```
cost = c(.001, .01, .1, 1)
epsilon = c(.001, .01, .1, 1)
set.seed(4645251)
tune.out = tune(svm, medv ~ ., data=Boston[train,], kernel='linear'
                , ranges=list(cost=cost, epsilon=epsilon))

perf = tune.out$performances
perf$cost = factor(perf$cost)
ggplot(perf) + geom_line(aes(epsilon, error, group=cost, color=cost)) + scale_x_log10()
```




```
tune.out$best.parameters
```

```
##   cost epsilon
## 4    1   0.001
```

```
tune.out$performances
```

```
##     cost epsilon    error dispersion
## 1  0.001   0.001 43.29272  13.078277
## 2  0.010   0.001 26.95169  10.557590
## 3  0.100   0.001 25.60440  10.829528
## 4  1.000   0.001 25.23923  10.888296
## 5  0.001   0.010 43.30323  13.053327
## 6  0.010   0.010 26.97850  10.603666
## 7  0.100   0.010 25.52956  10.792008
## 8  1.000   0.010 25.37565  11.032447
## 9  0.001   0.100 43.28249  12.821787
## 10 0.010   0.100 26.38916  10.443813
## 11 0.100   0.100 25.50024  10.688779
## 12 1.000   0.100 25.37825  10.714303
## 13 0.001   1.000 56.78102   9.859582
## 14 0.010   1.000 44.25528   7.906997
## 15 0.100   1.000 32.87987   8.671015
## 16 1.000   1.000 30.23543   6.873077
```

The best parameter tuple (cost, epsilon) from this 10-fold cross-validation training is (1, 0.001). The corresponding purple curve is convex throughout the entire range. The detailed performance results were also printed: we see that ($cost = 1, \epsilon = 0.001$) indeed has the lowest error, closely followed by ($cost = 1, \epsilon = 0.01$). The ($cost = 0.001, \epsilon = 0.001$) pair produced the highest error. Ultimately, a very low margin of tolerance and a high cost yields the lowest cross-validation MSE.

```
best.model = tune.out$best.model
ypred = predict(best.model, Boston[test,])
ytrue = Boston$medv[test]

mse = mean((ytrue - ypred)^2)
mae = mean(abs(ytrue - ypred))
cat('MSE:', round(mse, 4), '\n')
```

```
## MSE: 26.2043
```

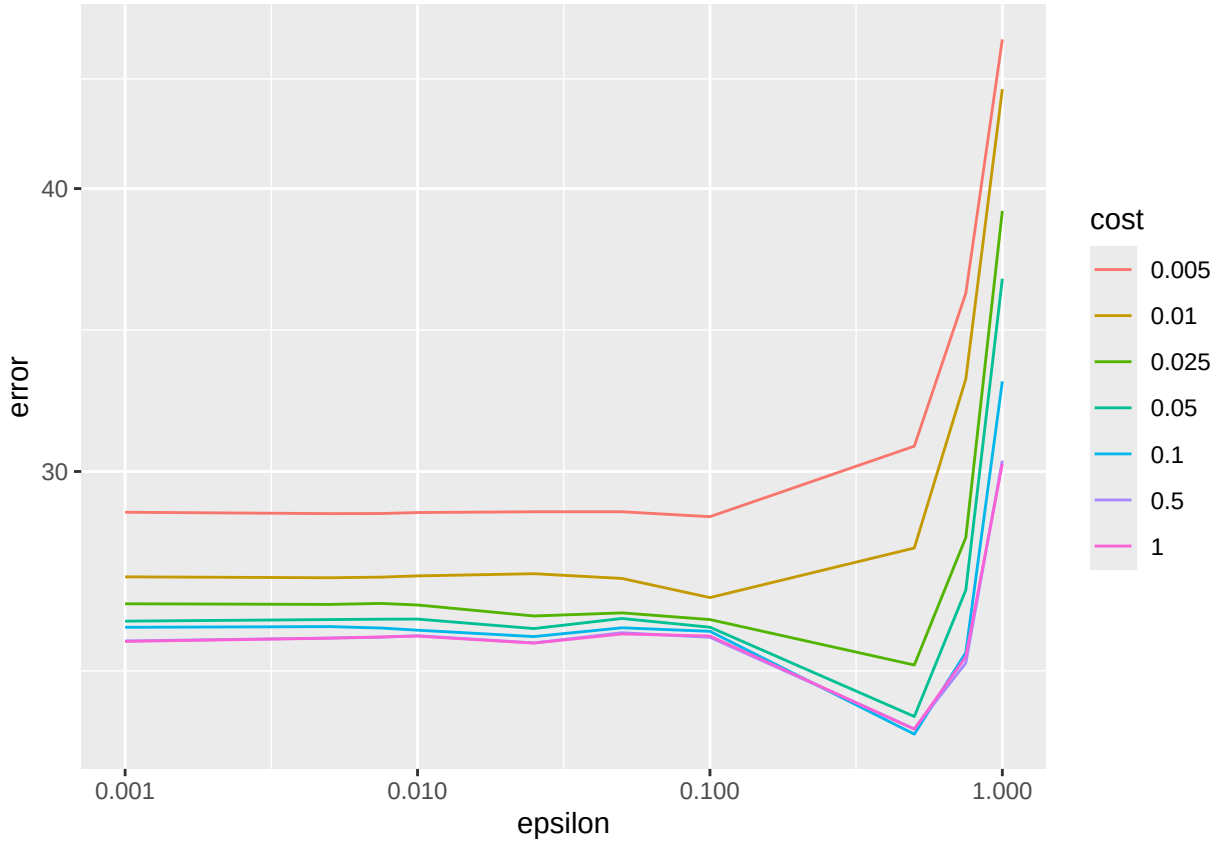
```
cat('MAE:', round(mae, 4))
```

```
## MAE: 3.5494
```

Let us now try a grid search with smaller resolution and compare the test MSEs:

```
cost = c(.005, .01, .025, .05, .1, .5, 1)
epsilon = c(.001, .005, .0075, .01, .025, .05, .1, .5, .75, 1)
set.seed(4645251)
tune.out = tune(svm, medv ~ ., data=Boston[train,], kernel='linear'
                , ranges=list(cost=cost, epsilon=epsilon))

perf = tune.out$performances
perf$cost = factor(perf$cost)
ggplot(perf) + geom_line(aes(epsilon, error, group=cost, color=cost)) + scale_x_log10() + scale_y_log10
```




```
tune.out$best.parameters
```

```
##    cost epsilon
## 54  0.1     0.5
```

After carrying out a more elaborate grid search, the best parameter tuple turns out to be ($cost = 0.1$, $\epsilon = 0.5$). This is a much more balanced pair, with a much higher margin of tolerance and softer penalisation outside of the margin. The cross-validation MSE obtained with this pair is 22.96781, which is lower than the previous best training MSE from the higher resolution grid.

```
best.model = tune.out$best.model
ypred = predict(best.model, Boston[test,])
ytrue = Boston$medv[test]

mse = mean((ytrue - ypred)^2)
mae = mean(abs(ytrue - ypred))
cat('MSE:', round(mse, 4), '\n')
```

```
## MSE: 25.0509
```

```
cat('MAE:', round(mae, 4))
```

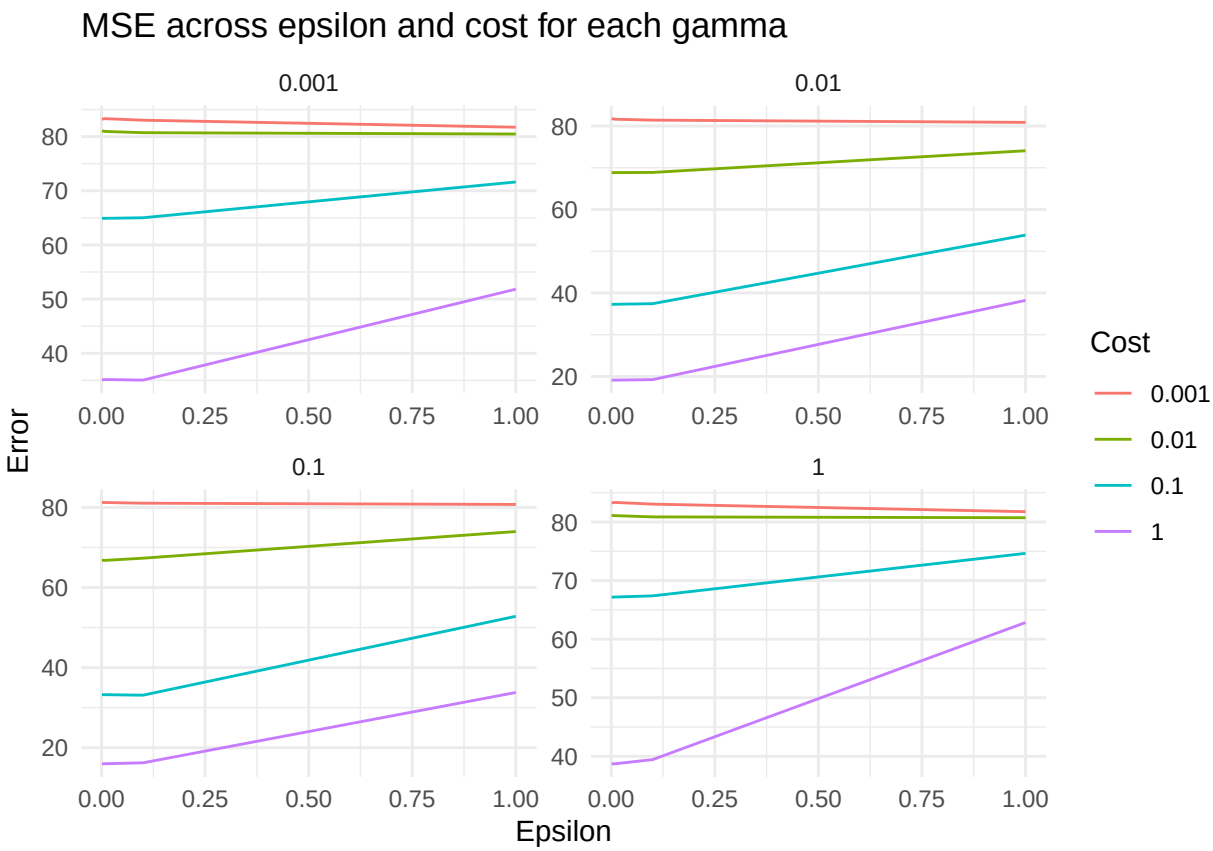
```
## MAE: 3.6301
```

And with this pair we also obtain a lower test MSE - almost 5% lower, which is not a huge gain. Nevertheless, it is interesting to note that the mean absolute error is slightly higher for this pair than for the previous best pair - 2% higher. The previous higher cost parameter likely lead to a little bit of overfitting.

```
cost = c(.001, .01, .1, 1)
epsilon = c(.001, .01, .1, 1)
gamma = c(.001, .01, .1, 1)
set.seed(4645251)
tune.out = tune(svm, medv ~ ., data=Boston[train,], kernel='radial'
                , ranges=list(cost=cost, epsilon=epsilon, gamma=gamma))

perf = tune.out$performances
perf$cost = factor(perf$cost)
res <- tune.out$performances

# Plot: one for each gamma
ggplot(perf, aes(epsilon, error, group=cost, color=cost)) +
  geom_line() +
  facet_wrap(~ gamma, scales = "free") +
  labs(
    title = "MSE across epsilon and cost for each gamma",
    x = "Epsilon",
    y = "Error",
    color = "Cost"
  ) +
  theme_minimal()
```



```
tune.out$best.parameters
```

```
##    cost epsilon gamma
## 36    1   0.001   0.1
```

After performing cross-validation with an SVR with a radial basis kernel, we find that the best parameter triplet is ($cost = 1$, $\epsilon = 0.001$, $\gamma = 0.1$). The first two values are familiar, since they are the exact same obtained with a linear kernel and the same resolution grid in our first experiment.

```
best.model = tune.out$best.model
ypred = predict(best.model, Boston[test,])
ytrue = Boston$medv[test]

mse = mean((ytrue - ypred)^2)
mae = mean(abs(ytrue - ypred))
cat('MSE:', round(mse, 4), '\n')
```

```
## MSE: 17.1149
```

```
cat('MAE:', round(mae, 4))
```

```
## MAE: 2.442
```

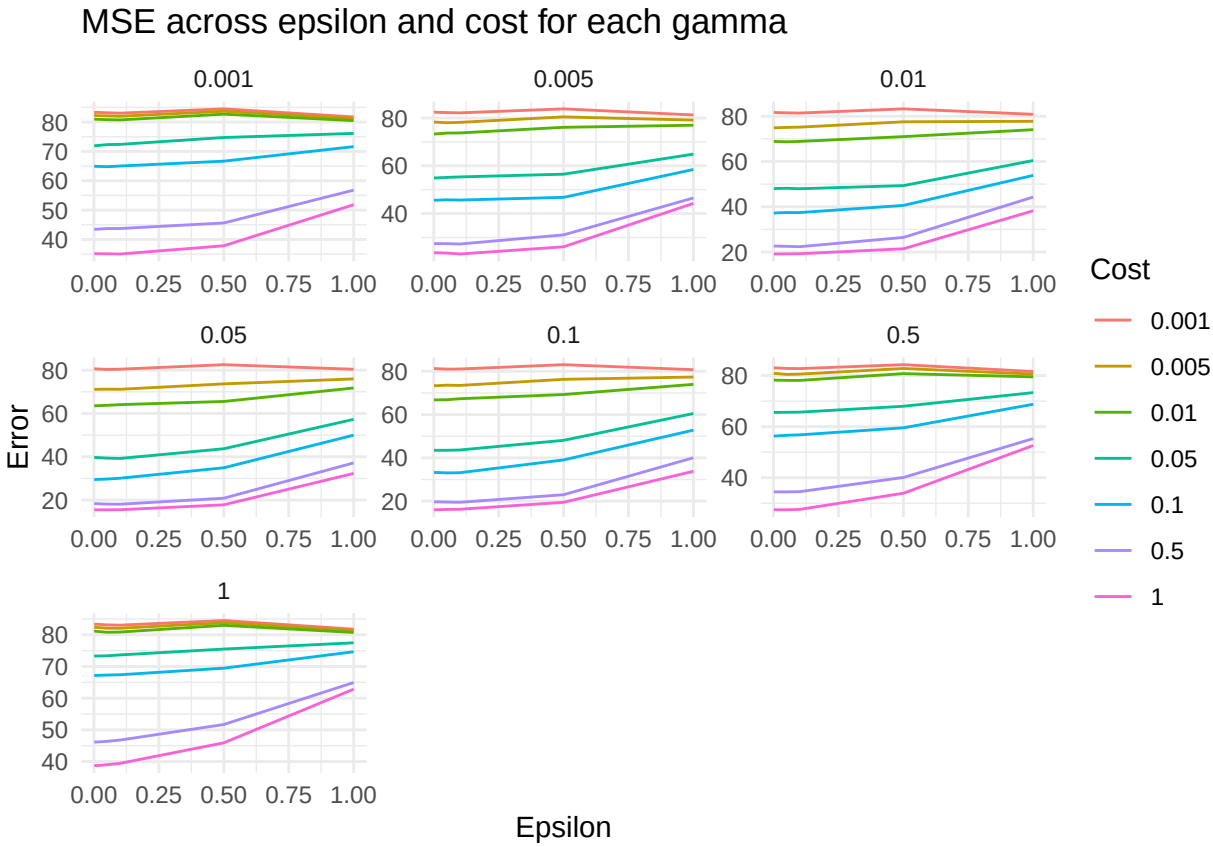
Both the MSE and MAE are significantly lower than what our best parameter pair yielded with a linear kernel.

Let us now repeat this experiment with a higher resolution grid:

```
cost = c(.001, .005, .01, .05, .1, .5, 1)
epsilon = c(.001, .005, .01, .05, .1, .5, 1)
gamma = c(.001, .005, .01, .05, .1, .5, 1)
set.seed(4645251)
tune.out = tune(svm, medv ~ ., data=Boston[train,], kernel='radial'
                , ranges=list(cost=cost, epsilon=epsilon, gamma=gamma))

perf = tune.out$performances
perf$cost = factor(perf$cost)
res <- tune.out$performances

# Plot: one for each gamma
ggplot(perf, aes(epsilon, error, group=cost, color=cost)) +
  geom_line() +
  facet_wrap(~ gamma, scales = "free") +
  labs(
    title = "MSE across epsilon and cost for each gamma",
    x = "Epsilon",
    y = "Error",
    color = "Cost"
  ) +
  theme_minimal()
```




```
tune.out$best.parameters
```

```
##     cost epsilon gamma
## 175    1    0.05  0.05
```

```
best.model = tune.out$best.model
ypred = predict(best.model, Boston[test,])
ytrue = Boston$medv[test]

mse = mean((ytrue - ypred)^2)
mae = mean(abs(ytrue - ypred))
cat('MSE:', round(mse, 4), '\n')
```

```
## MSE: 16.8298
```

```
cat('MAE:', round(mae, 4))
```

```
## MAE: 2.4546
```

It turns out that the ($cost = 1$, $\epsilon = 0.05$, $\gamma = 0.05$) triplet gives an even lower MSE - though the gain is really minimal, only 1.5% -, but offers no improvement in terms of MAE. The fact that the best gamma is relatively high means that each point influences a rather small region, so bias is lower here. Our training set is relatively big in statistical terms, so it is no surprise that lowering the bias yields better predictions.

```r
lm.model <- lm(medv ~ ., data = Boston[train, ])
lm.pred <- predict(lm.model, newdata = Boston[test, ])

mse = mean((ytrue - lm.pred)^2)
mae = mean(abs(ytrue - lm.pred))
cat('MSE:', round(mse, 4), '\n')
```

```
## MSE: 26.151
```

```r
cat('MAE:', round(mae, 4))
```

```
## MAE: 3.7217
```

The error reduction compared to a simple multivariate regression is indeed significant both in terms of MSE and MAE.