R 机器学习

第 08 讲　KNN

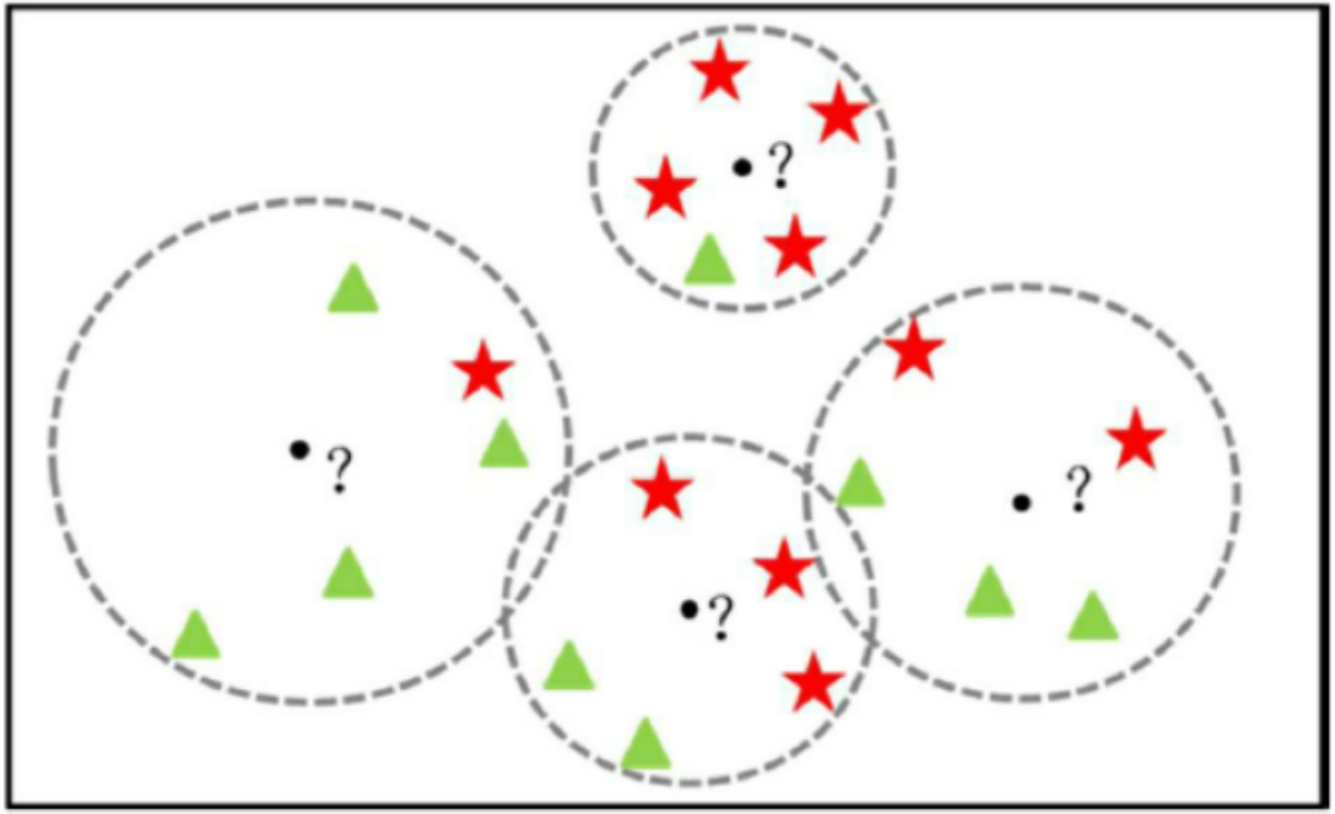
张敬信

2022 年 10 月 4 日

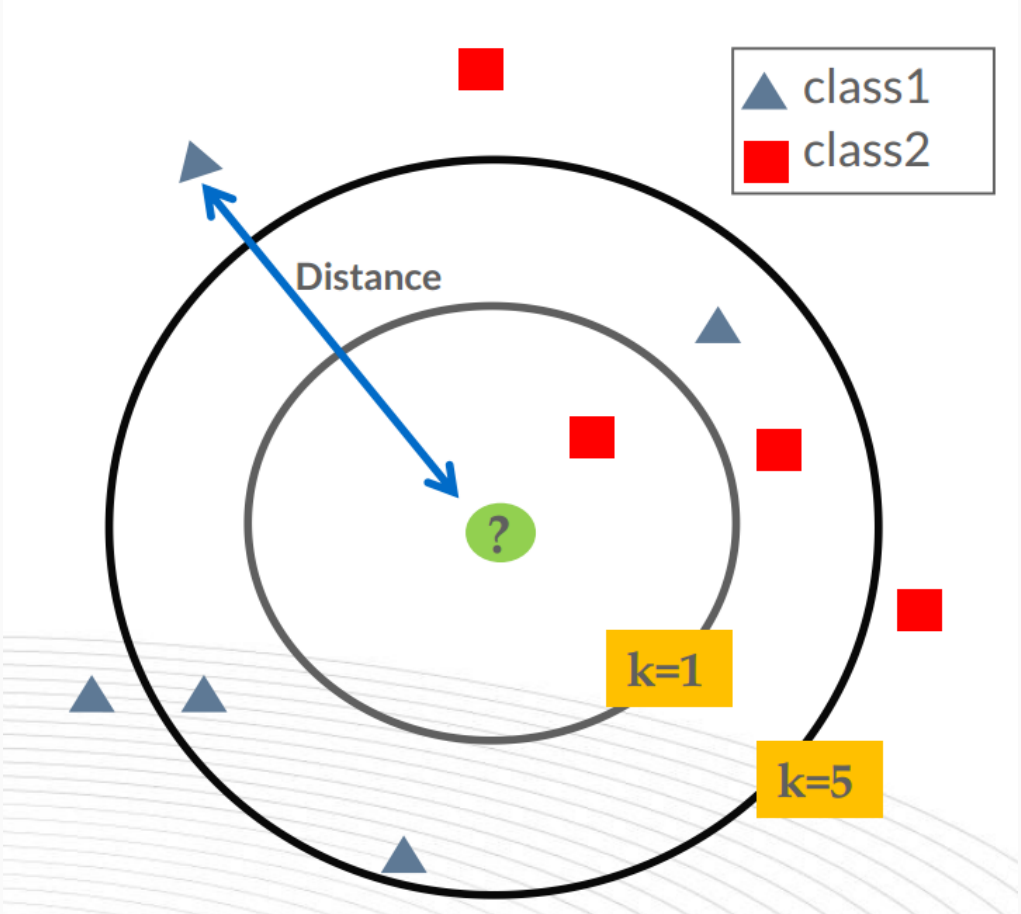
哈尔滨商业大学

一. KNN 原理

- K 近邻（KNN）是简单且特别的数据挖掘算法，它不像其它模型有损失函数、有优化算法、有训练过程。KNN 既可以作分类，也可以作回归。
- KNN 可以概况为：“**近朱者赤，近墨者黑**”，具体是通过搜索最近的 K 个已知类别样本用于未知类别样本的预测。“最近” 的度量就是用点之间的距离或相似性。距离越小或相似度越高，说明它们之间越近。
- KNN **分类**：针对离散型的因变量，一个样本的类别是由其邻居的” 多数表决（投票）“确定的，K 个最近邻居中出现最多的类别就是该样本的预测类别；
- KNN **回归**：针对连续型的因变量，则是将 K 个最近邻居的因变量的均值用作该样本的因变量预测值。



- KNN 示意图，表示样本有两种类别：五角星和三角形，选择邻居数 $K = 5$，通过投票的方式就能确定未知样本的类别。



- 可见，不同的 K 值，可能产生完全不同的预测结果：$K = 1$ 预测为 Class2，$K = 5$ 预测为 Class1.

KNN 基本步骤

(1) 计算待分类样本与其它样本之间的距离；

(2) 统计距离最近的 K 个邻居；

(3) 对于 K 个最近的邻居，分类则看它们属于哪个分类最多，待分类样本就属于哪一类；回归则计算它们的因变量的均值，作为待回归样本的预测值。

这就引出两个问题:

- 怎么度量距离？(不同的距离，得到的 K 个邻居可能不同)
- 如何确定最优的 K 值？(过小会过拟合，过大会欠拟合)

1. 常用的距离度量

对于样本点 $A(x_1, \ldots, x_m)$, $B(y_1, \ldots, y_m)$.

- **欧氏距离**:

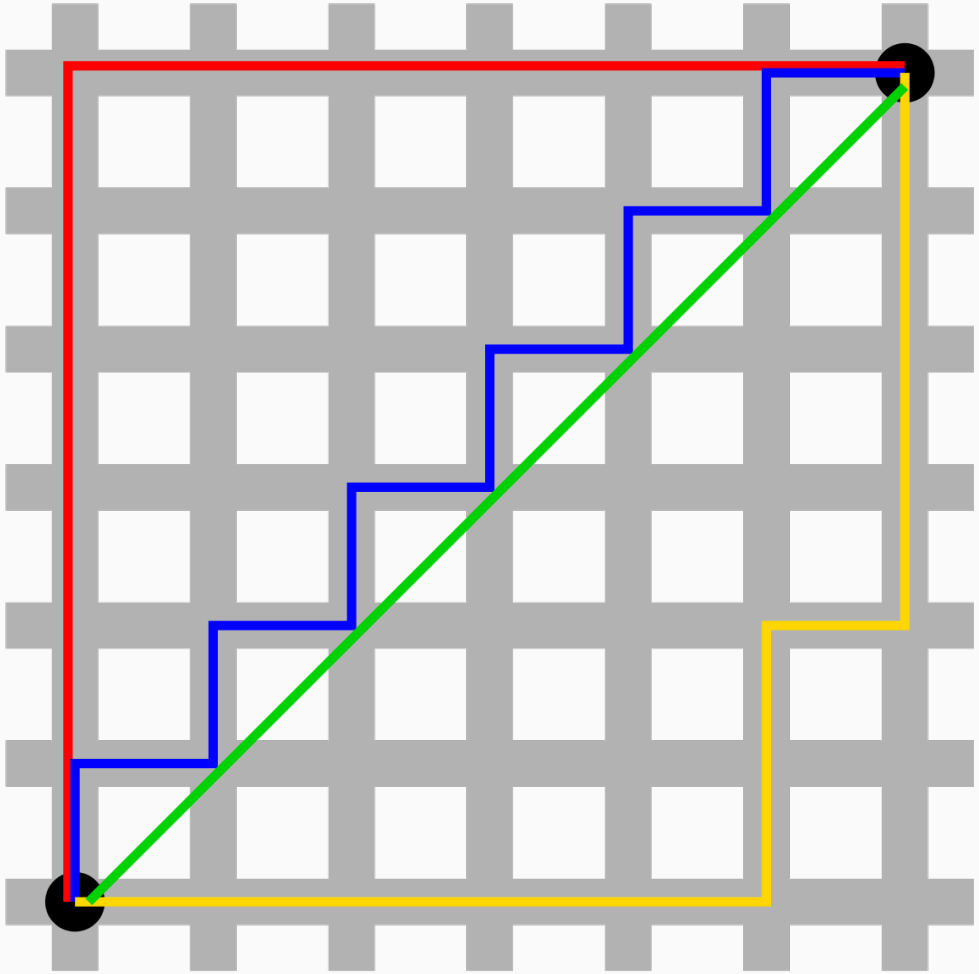
$$d(A, B) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$$

即通常的两点间直线距离的推广。

- **曼哈顿距离**:

$$d_1(A, B) = \sum_{i=1}^{m}|x_i - y_i|$$

也称为" 街区距离"，沿直角直线过去。



注：更一般的是闵科夫斯基距离，取 $p = 2$ 为欧氏距离，$p = 1$ 为曼哈顿距离，$p = +\infty$ 为切比雪夫距离。

- **余弦相似度**

$$\cos(A,\ B) = \frac{\langle A,\ B\rangle}{\|A\| \cdot \|B\|}$$

余弦相似度，即两个向量的夹角余弦，可以理解为方向一致程度。

- **杰卡德相似系数**：

$$J(A,\ B) = \frac{|A \cap B|}{|A \cup B|}$$

表示两个集合交集与并集元素个数之比。

注：余弦相似度与杰卡德相似系数，常用于文本挖掘和推荐算法，用来度量文本数据和用户的相似性

- 使用距离度量样本间的相似性时要注意：数值型变量的量纲影响，需要做标准化/归一化；离散型变量需要转化为哑变量或数值化表示，才能考虑距离。
- 其它距离还有很多，比如不受量纲影响的马氏（mahalanobis）距离、度量字符串距离的汉明距离和编辑距离、与信息熵有关的 KL 散度（相对熵)、互信息，等。
- `philentropy` 包实现了 46 种距离的计算。

2. 最优 K 值的选择

- **交叉验证法**：取不同的 K 值，然后在每个 K 值下执行 k 折交叉验证，最后选出分类错误率或平均误差最小的 K 值。
- **近邻样本加权**：若已知样本距离未知样本比较远，则对应的权重就设置得低一些，否则权重就高一些，比如设为距离的倒数。

3. 近邻样本的搜索方法

- KNN 的计算过程是大量计算样本点之间的距离。为了减少计算距离的次数，提升 KNN 的搜索效率，提出了 KD 树和球树算法。
- KD 树是对数据点在 K 维空间中划分的一种数据结构。在 KD 树的构造中，每个节点都是 K 维数值点的二叉树，从而可以用二叉树的增删改查操作，这样就大大提升了搜索效率。
- KD 树实际上是按照 K 维的坐标轴对数据进行划分，最终将 K 维空间切割为一个个超矩体（包含叶结点），当未知类别的样本进入 KD 树后，就会自顶向下地流淌到对应的叶结点中，并反向计算最近邻样本。
- KD 树的超矩体有”角”容易产生模棱两可的区域，将超矩体换成超球体，就是球树算法。

二. KNN 实例：分类

- 以来自 UCI 的 Knowledge 数据集为例，是关于学生知识掌握的指标数据，包含 403 个观测，6 个变量。
- 自变量包括：STG（目标科目学习时长）、SCG（重复次数）、STR（相关科目学习时长）、LPR（相关科目考试成绩）、PEG（目标科目考试成绩），因变量为 UNS（知识掌握程度）。
- 自变量数据是归一化后的数据，因变量是知识掌握程度，是分类变量。
- 本例将阐述 KNN 分类的一般流程，同时系统地学习一下超参数调参技术。

1. 准备数据

```
library(tidyverse)
library(mlr3verse)

dat = readxl::read_xlsx("datas/Knowledge.xlsx")
dat$UNS = as.factor(dat$UNS)
head(dat, 2)
#> # A tibble: 2 x 6
#>     STG   SCG   STR   LPR   PEG UNS
#>   <dbl> <dbl> <dbl> <dbl> <dbl> <fct>
#> 1  0     0     0     0     0   Very Low
#> 2  0.08  0.08  0.1   0.24  0.9 High
table(dat$UNS)
#>
#>     High      Low   Middle Very Low
#>      102      129      122       50
```

2. 创建任务

```
task = as_task_classif(dat, target = "UNS")
task
#> <TaskClassif:dat> (403 x 6)
#> * Target: UNS
#> * Properties: multiclass
#> * Features (5):
#>   - dbl (5): LPR, PEG, SCG, STG, STR
```

3. 选择学习器

```
# 需要 kknn 包，选择不加权 KNN
knn = lrn("classif.kknn", predict_type = "prob",
          kernel = "rectangular", scale = FALSE)
knn
#> <LearnerClassifKKNN:classif.kknn>
#> * Model: -
#> * Parameters: k=7, kernel=rectangular, scale=FALSE
#> * Packages: mlr3, mlr3learners, kknn
#> * Predict Types:  response, [prob]
#> * Feature Types: logical, integer, numeric, factor, ordere
#> * Properties: multiclass, twoclass
```

4. 划分训练集测试集

- 做留出 (holdout) 重抽样，70% 作为训练集，其余 30% 作为测试集
- 为了保持训练集、测试集的因变量数据具有相似的分布，采用分层抽样方法
- 用 partition( ) 函数对任务做划分，默认按因变量分层，取出训练集索引和测试集索引

```
set.seed(123)
split = partition(task, ratio = 0.7)
# 默认 stratify = TRUE
```

5. 超参数调参 I（简单调参）

- 查看模型的所有超参数及默认值

```
knn$param_set
#> <ParamSet>
#>             id    class lower upper nlevels default
#> 1:           k ParamInt     1   Inf     Inf       7
#> 2:    distance ParamDbl     0   Inf     Inf       2
#> 3:      kernel ParamFct    NA    NA      10 optimal rectar
#> 4:       scale ParamLgl    NA    NA       2    TRUE
#> 5:     ykernel ParamUty    NA    NA     Inf
#> 6: store_model ParamLgl    NA    NA       2   FALSE
```

- 对模型中超参数：邻居数 k, 距离 distance (对应 p) 做调参
- 使用自动调参器，需要设置学习器、重抽样方法、模型评估指标、搜索空间、终止条件、搜索方法

```
library(paradox)
search_space = ps(
  k = p_int(lower = 2, upper = 15),
  distance = p_int(lower = 1, upper = 2))

at = auto_tuner(
  learner = knn,
  resampling = rsmp("cv", folds = 10),
  measure = msr("classif.acc"),
  search_space = search_space,
  method = "grid_search",
  term_evals = 10)
```

- 在训练集上启动调参过程：

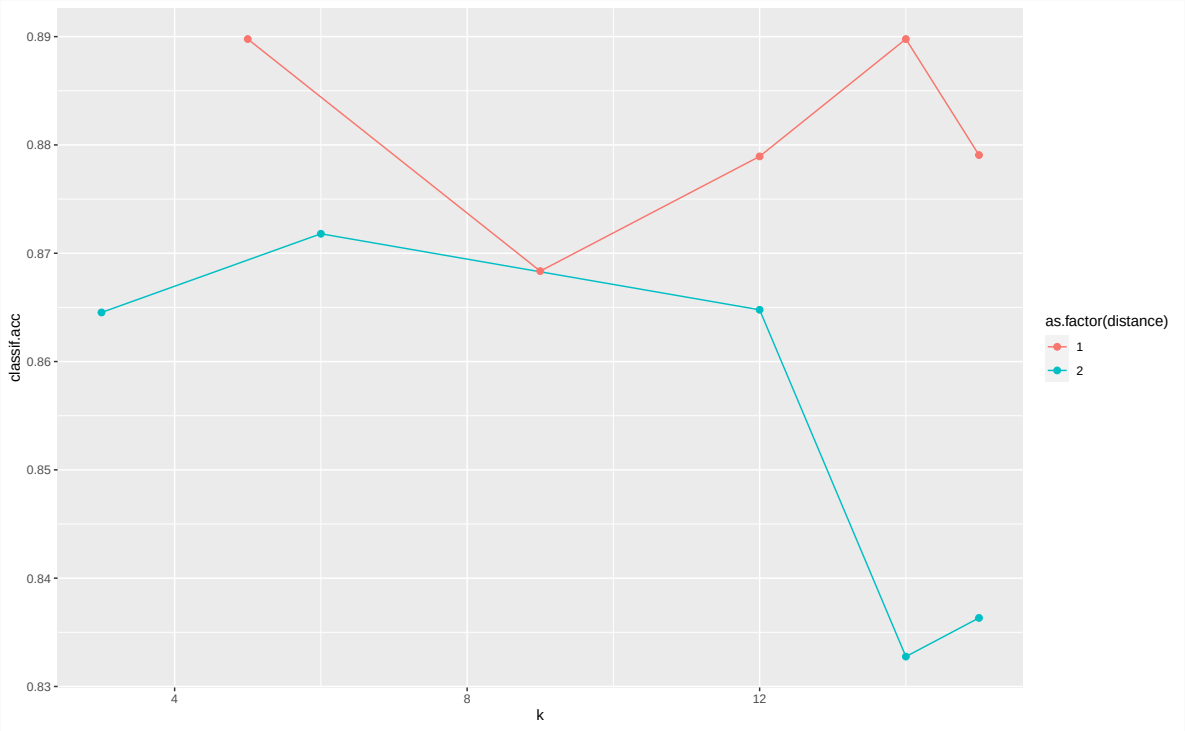
```
at$train(task, row_ids = split$train)
#> INFO  [17:43:01.392] [bbotk] Starting to optimize 2 parame
#> INFO  [17:43:01.423] [bbotk] Evaluating 1 configuration(s)
#> INFO  [17:43:01.442] [mlr3] Running benchmark with 10 resa
#> INFO  [17:43:01.502] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:01.535] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:01.570] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:01.597] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:01.624] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:01.653] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:01.680] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:01.708] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:01.737] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:01.764] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:01.795] [mlr3] Finished benchmark
#> INFO  [17:43:01.846] [bbotk] Result of batch 1:
```

- 查看最优超参数

```
at$tuning_result
#>    k distance learner_param_vals  x_domain classif.acc
#> 1: 5        1          <list[4]> <list[2]>        0.89
```

```
tunerlt = as.data.table(at$archive)
tunerlt %>%
  select(k, distance, classif.acc) %>%
  slice_max(classif.acc, n = 5)
#>     k distance classif.acc
#> 1:  5        1       0.890
#> 2: 14        1       0.890
#> 3: 15        1       0.879
#> 4: 12        1       0.879
#> 5:  6        2       0.872
```

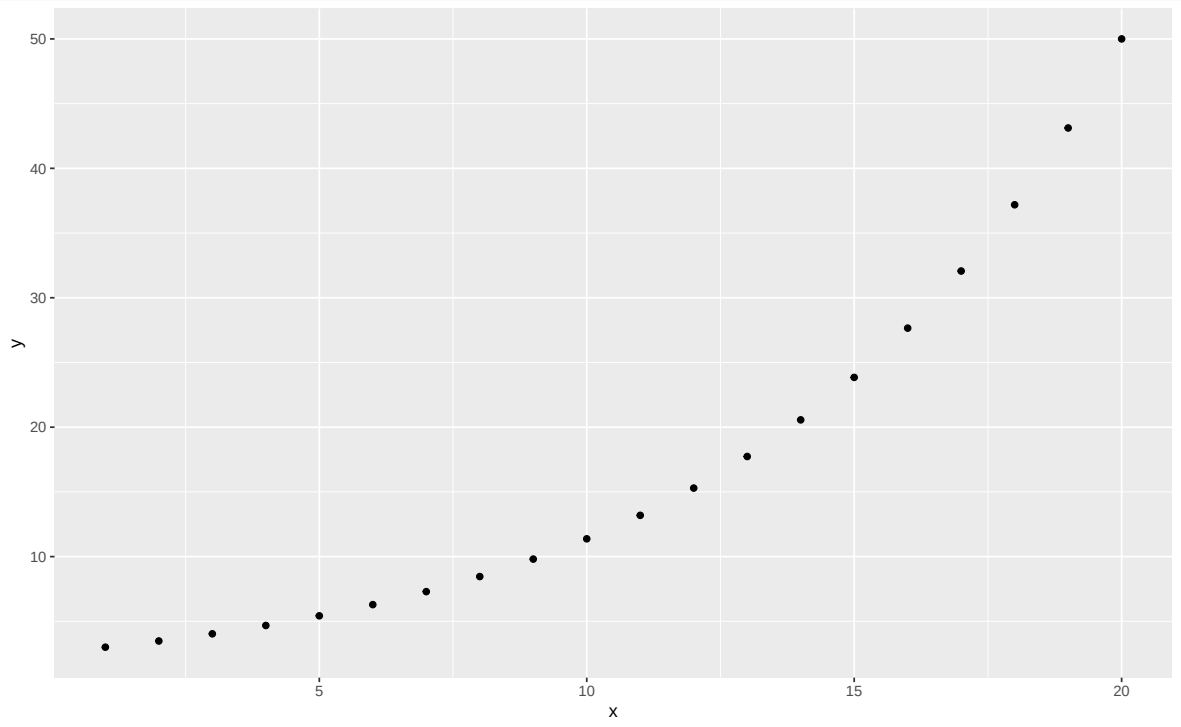
```
ggplot(tunerlt, aes(k, classif.acc,
                    color = as.factor(distance))) +
  geom_line() + geom_point(size = 2)
```



6. 超参数调参 II（随机搜索与超参数变换）

- k = 3 与 4 的差异要比 k = 100 与 101 的更大，所以在调参时，希望前期的点比后期的点更密集一些。这可以通过**对数-指数**变换来实现，这也适用于大范围搜索空间。

```
tibble(x = 1:20,
       y = exp(seq(log(3), log(50), length.out=20))) %>%
  ggplot(aes(x, y)) + geom_point()
```



- 对 k 不是均匀调参，而是加入**对数-指数**变换，同时再增加对 kernel 调参：

```
search_space = ps(
  k = p_dbl(lower = log(3), upper = log(30),
            trafo = function(x) round(exp(x))),
  distance = p_int(lower = 1, upper = 2),
  kernel = p_fct(levels = c("rectangular", "gaussian",
                            "rank", "optimal")))
```

```
at = auto_tuner(
  learner = knn,
  resampling = rsmp("cv", folds = 10),
  measure = msr("classif.acc"),
  search_space = search_space,
  method = "random_search",
  term_evals = 20)
```

- 在训练集上启动调参过程

```
at$train(task, row_ids = split$train)
#> INFO  [17:43:05.771] [bbotk] Starting to optimize 3 parame
#> INFO  [17:43:05.792] [bbotk] Evaluating 1 configuration(s)
#> INFO  [17:43:05.804] [mlr3] Running benchmark with 10 resa
#> INFO  [17:43:05.810] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:05.839] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:05.869] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:05.895] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:05.929] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:05.956] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:05.985] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:06.011] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:06.036] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:06.065] [mlr3] Applying learner 'classif.kknn
#> INFO  [17:43:06.091] [mlr3] Finished benchmark
#> INFO  [17:43:06.155] [bbotk] Result of batch 1:
```

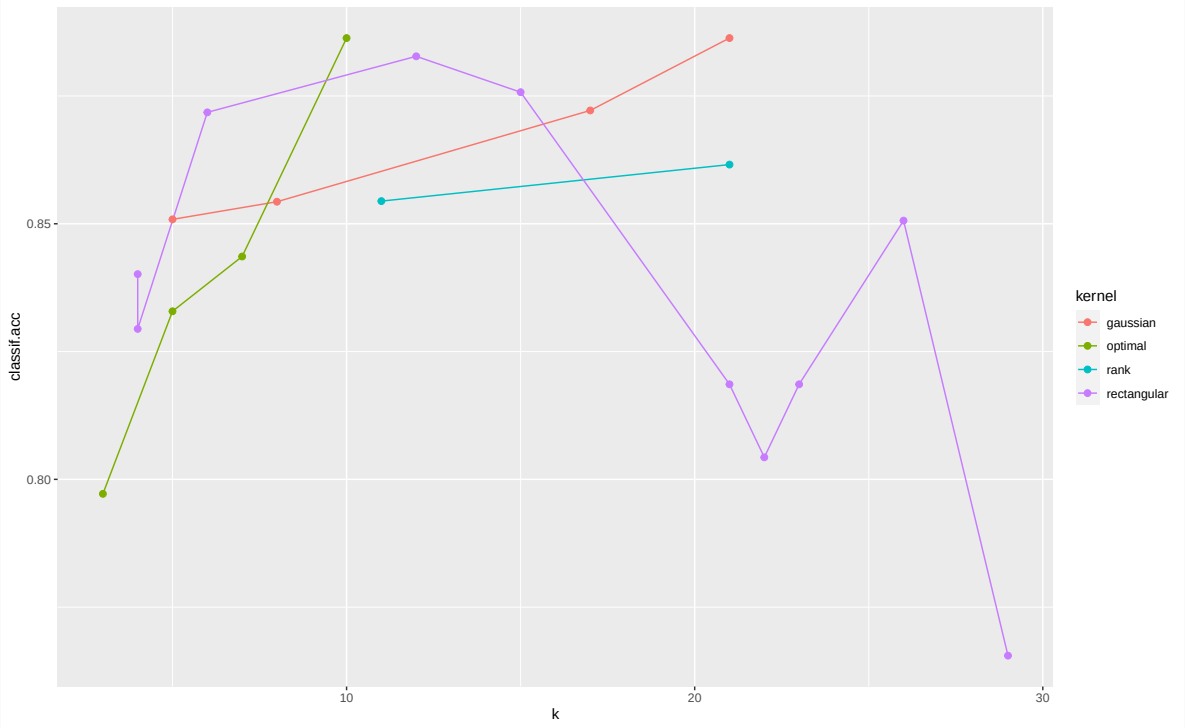
- 查看最优超参数

```
at$tuning_result
#>       k distance  kernel learner_param_vals  x_domain clas
#> 1: 2.27        1 optimal          <list[4]> <list[3]>
```

- 展示更多调参结果

```
tunerlt = as.data.table(at$archive) %>%
  select(k = x_domain_k, distance, kernel, classif.acc)
tunerlt %>%
  slice_max(classif.acc, n = 5)
#>     k distance      kernel classif.acc
#> 1: 10        1     optimal       0.886
#> 2: 21        1    gaussian       0.886
#> 3: 12        1 rectangular       0.883
#> 4: 15        1 rectangular       0.876
#> 5: 17        2    gaussian       0.872
```

```
ggplot(tunerlt, aes(k, classif.acc, color = kernel)) +
  geom_line() + geom_point(size = 2)
```



7. 训练模型

- 用调出的最优参数更新学习器的参数集，然后训练模型

```
knn$param_set$values = at$tuning_result$learner_param_vals[[1
knn$train(task, row_ids = split$train)
```

8. 模型预测及评估

```
prediction = knn$predict(task, row_ids = split$test)

prediction$confusion                    # 混淆矩阵
#>            truth
#> response    High Low Middle Very Low
#>    High       31   0      0         0
#>    Low         0  38      3         2
#>    Middle      0   1     34         0
#>    Very Low    0   0      0        13
prediction$score(msr("classif.acc"))    # 准确率
#> classif.acc
#>       0.951
```

9. 预测新数据

```
newdata = dat[1:5,-6]
knn$predict_newdata(newdata)
#> <PredictionClassif> for 5 observations:
#>  row_ids truth response prob.High prob.Low prob.Middle pro
#>        1  <NA> Very Low     0.000    0.264        0.000
#>        2  <NA>     High     0.455    0.187        0.358
#>        3  <NA>      Low     0.000    0.904        0.000
#>        4  <NA>   Middle     0.000    0.500        0.500
#>        5  <NA>   Middle     0.000    0.459        0.541
```

三. KNN 实例：回归

- 以来自 UCI 的 CCPP 数据集为例，是关于高炉煤气联合循环发电的指标数据，包含 9568 个观测，5 个变量，均为连续型。
- 自变量包括：AT（高炉温度）、V（炉内压力）、AP（相对湿度）、RH（高炉排气量）；因变量为 PE（高炉发电量）。
- 数据量纲不同，需要做归一化。
- 本例将阐述 KNN 回归的一般流程。

1. 准备数据

```
library(tidyverse)
library(mlr3verse)

dat = readxl::read_xlsx("datas/CCPP.xlsx")
head(dat)
#> # A tibble: 6 x 5
#>      AT     V    AP    RH    PE
#>   <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1 15.0   41.8 1024.  73.2  463.
#> 2 25.2   63.0 1020.  59.1  444.
#> 3  5.11  39.4 1012.  92.1  489.
#> 4 20.9   57.3 1010.  76.6  446.
#> 5 10.8   37.5 1009.  96.6  474.
#> 6 26.3   59.4 1012.  58.8  444.
```

2. 创建任务

```
task = as_task_regr(dat, target = "PE")
task
#> <TaskRegr:dat> (9568 x 5)
#> * Target: PE
#> * Properties: -
#> * Features (4):
#>   - dbl (4): AP, AT, RH, V
```

3. 选择学习器

```
# 需要 kknn 包, 选择不加权 KNN
knn = lrn("regr.kknn", scale = TRUE)
knn
#> <LearnerRegrKKNN:regr.kknn>
#> * Model: -
#> * Parameters: k=7, scale=TRUE
#> * Packages: mlr3, mlr3learners, kknn
#> * Predict Types:  [response]
#> * Feature Types: logical, integer, numeric, factor, ordere
#> * Properties: -
```

4. 划分训练集测试集

- 做留出 (holdout) 重抽样，70% 作为训练集，其余 30% 作为测试集
- 为了保持训练集、测试集的因变量数据具有相似的分布，采用分层抽样方法
- 用 partition( ) 函数对任务做划分，默认按因变量分层，取出训练集索引和测试集索引

```
set.seed(123)
split = partition(task, ratio = 0.7)
# 默认 stratify = TRUE
```

5. 超参数调参

- 查看模型的所有超参数及默认值

```
knn$param_set
#> <ParamSet>
#>             id    class lower upper nlevels default value
#> 1:           k ParamInt     1   Inf     Inf       7     7
#> 2:    distance ParamDbl     0   Inf     Inf       2
#> 3:      kernel ParamFct    NA    NA      10 optimal
#> 4:       scale ParamLgl    NA    NA       2    TRUE  TRUE
#> 5:     ykernel ParamUty    NA    NA     Inf
#> 6: store_model ParamLgl    NA    NA       2   FALSE
```

- 对模型中超参数：邻居数 k, 距离 `distance` (对应 p), `kernel` 做调参
- 其中，对 k 不是均匀调参，而是加入**对数-指数**变换

```
search_space = ps(
  k = p_dbl(lower = log(3), upper = log(30),
            trafo = function(x) round(exp(x))),
  distance = p_int(lower = 1, upper = 3),
  kernel = p_fct(levels = c("rectangular", "gaussian",
                            "rank", "optimal")))
```

- 使用自动调参器，需要设置学习器、重抽样方法、模型评估指标、搜索空间、终止条件、搜索方法

```
at = auto_tuner(
  learner = knn,
  resampling = rsmp("cv", folds = 5),
  measure = msr("regr.rmse"),
  search_space = search_space,
  method = "random_search",
  term_evals = 10)
```

- 在训练集上启动调参过程

```
at$train(task, row_ids = split$train)
#> INFO  [17:43:13.737] [bbotk] Starting to optimize 3 parame
#> INFO  [17:43:13.752] [bbotk] Evaluating 1 configuration(s)
#> INFO  [17:43:13.769] [mlr3] Running benchmark with 5 resam
#> INFO  [17:43:13.783] [mlr3] Applying learner 'regr.kknn' o
#> INFO  [17:43:14.078] [mlr3] Applying learner 'regr.kknn' o
#> INFO  [17:43:14.365] [mlr3] Applying learner 'regr.kknn' o
#> INFO  [17:43:14.646] [mlr3] Applying learner 'regr.kknn' o
#> INFO  [17:43:14.925] [mlr3] Applying learner 'regr.kknn' o
#> INFO  [17:43:15.216] [mlr3] Finished benchmark
#> INFO  [17:43:15.248] [bbotk] Result of batch 1:
#> INFO  [17:43:15.249] [bbotk]     k distance      kernel re
#> INFO  [17:43:15.249] [bbotk]  2.18        1 rectangular
#> INFO  [17:43:15.249] [bbotk]
#> INFO  [17:43:15.249] [bbotk]  6569f53f-e725-48ae-a1f5-ec62
#> INFO  [17:43:15.253] [bbotk] Evaluating 1 configuration(s)
```

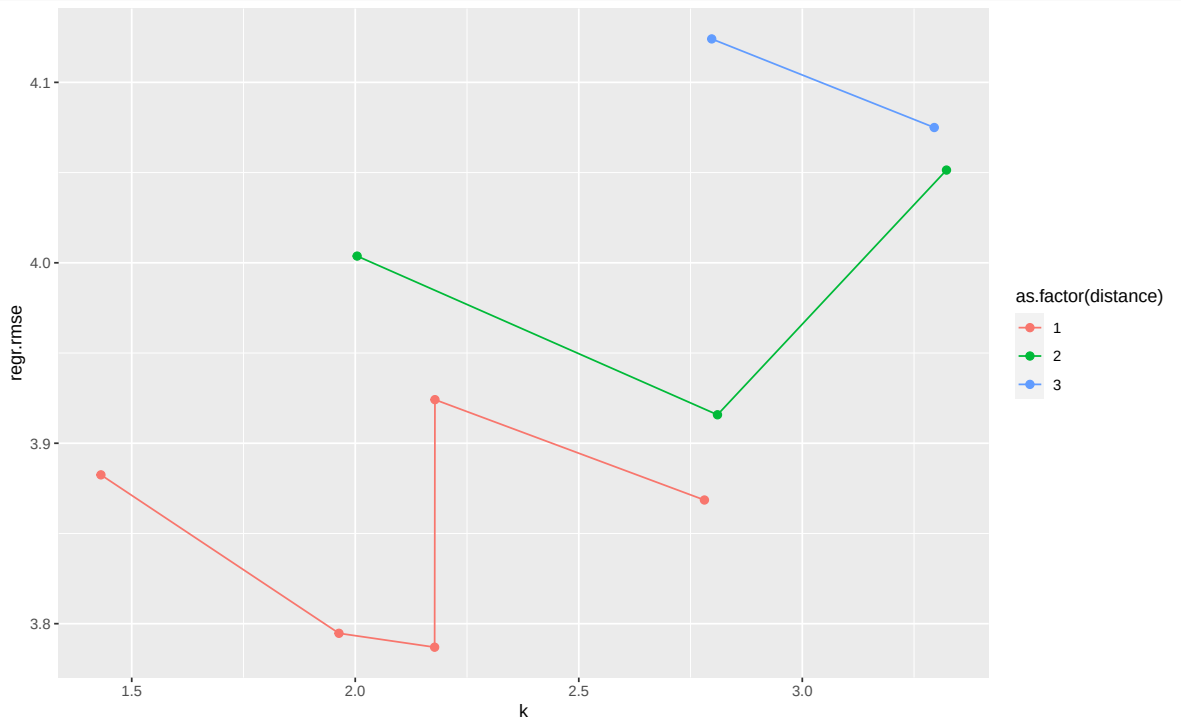
- 查看最优超参数

```
at$tuning_result
#>        k distance   kernel learner_param_vals  x_domain reg
#> 1: 2.18        1 gaussian          <list[4]> <list[3]>
```

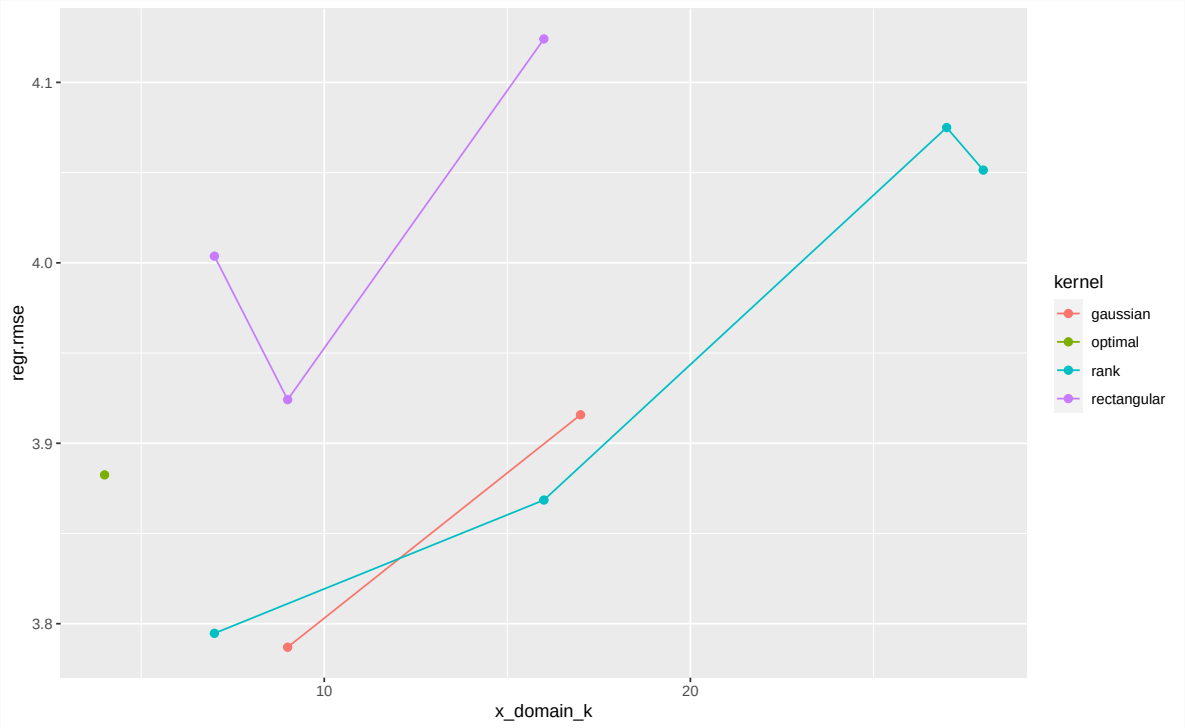
- 展示更多调参结果

```
tunerlt = as.data.table(at$archive)
tunerlt %>%
  select(k = x_domain_k, distance, kernel, regr.rmse) %>%
  arrange(regr.rmse) %>%
  slice(1:5)
#>     k distance   kernel regr.rmse
#> 1:  9        1 gaussian      3.79
#> 2:  7        1     rank      3.79
#> 3: 16        1     rank      3.87
#> 4:  4        1  optimal      3.88
#> 5: 17        2 gaussian      3.92
```

```
ggplot(tunerlt, aes(k, regr.rmse,
                    color = as.factor(distance))) +
  geom_line() + geom_point(size = 2)
```



```
ggplot(tunerlt, aes(x_domain_k, regr.rmse, color = kernel)) +
  geom_line() + geom_point(size = 2)
```



6. 训练模型

- 用调出的最优参数更新学习器的参数集

```
knn$param_set$values = at$tuning_result$learner_param_vals[[1
knn$train(task, row_ids = split$train)
```

7. 模型预测及评估

```r
prediction = knn$predict(task, row_ids = split$test)

prediction$score(msr("regr.rmse"))   # RMSE
#> regr.rmse
#>      3.65
prediction$score(msr("regr.rsq"))    # R 方
#> regr.rsq
#>    0.954
```

9. 预测新数据

```
newdata = dat[1:5,-5]
knn$predict_newdata(newdata)
#> <PredictionRegr> for 5 observations:
#>  row_ids truth response
#>        1    NA      462
#>        2    NA      444
#>        3    NA      489
#>        4    NA      446
#>        5    NA      474
```

主要参考文献

[1] mlr3book. 2020. https://mlr3book.mlr-org.com/

[2] Binder & Pfisterer (2020, March 11). mlr3gallery: mlr3tuning tutorial - german credit. https://mlr3gallery.mlr-org.com/posts/2020-03-11-mlr3tuning-tutorial-german-credit/.

[3] 刘顺祥. 从零开始学 Python: 数据分析与挖掘. 清华大学出版社, 2018.