



周志华《MACHINE LEARNING》学习笔记（17）-强化学习

👤 Admin ([Http://Spingspark.Com/Author/Admin/](http://Spingspark.Com/Author/Admin/)) 📦 框架

(<http://spingspark.com/category/%e6%a1%86%e6%9e%b6/>) 📅 七月 23, 2017

(<http://spingspark.com/%e6%a1%86%e6%9e%b6/%e5%91%a8%e5%bf%97%e5%8d%8e%e3%80%8a-machine-learning%e3%80%8b%e5%ad%a6%e4%b9%a0%e7%ac%94%e8%ae%b0%ef%bc%8817%ef%bc%89-%e5%bc%ba%e5%8c%96%e5%ad%a6%e4%b9%a0/>)

上篇主要介绍了概率图模型，首先从生成式模型与判别式模型的定义出发，引出了概率图模型的基本概念，即利用图结构来表达变量之间的依赖关系；接着分别介绍了隐马尔可夫模型、马尔可夫随机场、条件随机场、精确推断方法以及LDA话题模型；HMM主要围绕着评估/解码/学习这三个实际问题展开论述；MRF基于团和势函数的概念来定义联合概率分布；CRF引入两种特征函

数对状态序列进行评价打分；变量消去与信念传播在给定联合概率分布后计算特定变量的边际分布；LDA话题模型则试图去推断给定文档所蕴含的话题分布。本篇将介绍最后一种学习算法-强化学习。

16、强化学习

强化学习 (Reinforcement Learning, 简称RL) 是机器学习的一个重要分支, 前段时间人机大战的主角AlphaGo正是以强化学习为核心技术。在强化学习中, 包含两种基本的元素: **状态与动作**, **在某个状态下执行某种动作, 这便是一种策略**, 学习器要做的就是通过不断地探索学习, 从而获得一个好的策略。例如: 在围棋中, 一种落棋的局面就是一种状态, 若能知道每种局面下的最优落子动作, 那就攻无不克/百战不殆了~

若将状态看作为属性, 动作看作为标记, 易知: **监督学习和强化学习都是在试图寻找一个映射, 从已知属性/状态推断出标记/动作**, 这样强化学习中的策略相当于监督学习中的分类/回归器。但在实际问题中, **强化学习并没有监督学习那样的标记信息**, 通常都是在**尝试动作后才能获得结果**, 因此强化学习是通过反馈的结果信息不断调整之前的策略, 从而算法能够学习到: 在什么样的状态下选择什么样的动作可以获得最好的结果。

16.1 基本要素

强化学习任务通常使用**马尔可夫决策过程** (Markov Decision Process, 简称MDP) 来描述, 具体而言: 机器处在一个环境中, 每个状态为机器对当前环境的感知; 机器只能通过动作来影响环境, 当机器执行一个动作后, 会使得环境按某种概率转移到另一个状态; 同时, 环境会根据潜在的奖赏函数反馈给机器一个奖赏。综合而言, 强化学习主要包含四个要素: 状态、动作、转移概率以及奖赏函数。

状态 (X): 机器对环境的感知, 所有可能的状态称为状态空间;

动作 (A): 机器所采取的动作, 所有能采取的动作构成动作空间;

转移概率 (P): 当执行某个动作后, 当前状态会以某种概率转移到另一个状态;

奖赏函数 (R): 在状态转移的同时, 环境给反馈给机器一个奖赏。

仅探索法：将尝试的机会平均分给每一个动作，即轮流执行，最终将每个动作的平均奖赏作为期望奖赏的近似值。

仅利用法：将尝试的机会分给当前平均奖赏值最大的动作，隐含着让一部分人先富起来的思想。

可以看出：上述**两种方法是相互矛盾的**，仅探索法能较好地估算每个动作的期望奖赏，但是没能根据当前的反馈结果调整尝试策略；仅利用法在每次尝试之后都更新尝试策略，符合强化学习的思（tao）维（lu），但容易找不到最优动作。因此需要在这两者之间进行折中。

16.2.1 ϵ -贪心

ϵ -贪心法基于一个概率来对探索和利用进行折中，具体而言：在每次尝试时，以 ϵ 的概率进行探索，即以均匀概率随机选择一个动作；以 $1-\epsilon$ 的概率进行利用，即选择当前最优的动作。 ϵ -贪心法只需记录每个动作的当前平均奖赏值与被选中的次数，便可以增量式更新。

过程:

平均奖赏

选中次数

输出: 累积奖赏 r

Softmax算法则基于当前每个动作的平均奖赏值来对探索和利用进行折中，Softmax函数将一组值转化为一组概率，值越大对应的概率也越高，因此当前平均奖赏值越高的动作被选中的几率也越大。Softmax函数如下所示：

$$P(k) = \frac{e^{\frac{Q(k)}{\tau}}}{\sum_{i=1}^K e^{\frac{Q(i)}{\tau}}},$$

T : 温度
T->0:越放大差距
T->inf : 越缩小差距

T: 温度

T->0:越放大差距

T->inf : 越缩小差距

输入: 摇臂数 K ;
 奖赏函数 R ;
 尝试次数 T ;
 温度参数 τ .

过程:

```

1:  $r = 0$ ;
2:  $\forall i = 1, 2, \dots, K : Q(i) = 0, \text{count}(i) = 0$ ;
3: for  $t = 1, 2, \dots, T$  do
4:    $k =$  从  $1, 2, \dots, K$  中根据式(16.4)随机选取
5:    $v = R(k)$ ;
6:    $r = r + v$ ;
7:    $Q(k) = \frac{Q(k) \times \text{count}(k) + v}{\text{count}(k) + 1}$ ;
8:    $\text{count}(k) = \text{count}(k) + 1$ ;
9: end for

```

即根据Softmax函数

输出: 累积奖赏 r

<http://blog.csdn.net/u011826404>

16.3 有模型学习

若学习任务中的四个要素都已知，即状态空间、动作空间、转移概率以及奖赏函数都已经给出，这样的情形称为“**有模型学习**”。假设状态空间和动作空间均为有限，即均为离散值，这样我们不用通过尝试便可以对某个策略进行评估。

16.3.1 策略评估

前面提到：在模型已知的前提下，我们可以对任意策略的进行评估（后续会给出演算过程）。一般常使用以下两种值函数来评估某个策略的优劣：

状态值函数 (V) : $V(x)$ ，即从状态 x 出发，使用 π 策略所带来的累积奖赏；

状态-动作值函数 (Q)： $Q(x,a)$ ，即从状态 x 出发，执行动作 a 后再使用 π 策略所带来的累积奖赏。

根据累积奖赏的定义，我们可以引入T步累积奖赏与r折扣累积奖赏：

$$\begin{cases} Q_T^\pi(x, a) = \mathbb{E}_\pi[\frac{1}{T} \sum_{t=1}^T r_t \mid x_0 = x, a_0 = a]; \\ Q_\gamma^\pi(x, a) = \mathbb{E}_\pi[\sum_{t=0}^{+\infty} \gamma^t r_{t+1} \mid x_0 = x, a_0 = a]. \end{cases}$$

由于MDP具有马尔可夫性，即现在决定未来，将来和过去无关，我们很容易找到值函数的递归关系：

$$\begin{aligned}
 V_T^\pi(x) &= \mathbb{E}_\pi \left[\frac{1}{T} \sum_{t=1}^T r_t \mid x_0 = x \right] = \mathbb{E}_\pi \left[\frac{1}{T} r_1 + \frac{T-1}{T} \frac{1}{T-1} \sum_{t=2}^T r_t \mid x_0 = x \right] \\
 &= \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a \left(\frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} \mathbb{E}_\pi \left[\frac{1}{T-1} \sum_{t=1}^{T-1} r_t \mid x_0 = x' \right] \right) \quad \text{全概率展开} \\
 &= \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a \left(\frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V_{T-1}^\pi(x') \right) \quad \text{递归关系}
 \end{aligned}$$

类似地，对于r折扣累积奖赏可以得到：

$$V_{\gamma}^{\pi}(x) = \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a (R_{x \rightarrow x'}^a + \gamma V_{\gamma}^{\pi}(x')).$$

易知：当模型已知时，策略的评估问题转化为一种动态规划问题，即以填表格的形式自底向上，先求解每个状态的单步累积奖赏，再求解每个状态的两步累积奖赏，一直迭代逐步求解出每个状态的T步累积奖赏。算法流程如下所示：

$$\begin{cases} Q_T^*(x, a) = \sum_{x' \in X} P_{x \rightarrow x'}^a \left(\frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} \max_{a' \in A} Q_{T-1}^*(x', a') \right); \\ Q_\gamma^*(x, a) = \sum_{x' \in X} P_{x \rightarrow x'}^a (R_{x \rightarrow x'}^a + \gamma \max_{a' \in A} Q_\gamma^*(x', a')). \end{cases}$$

最优Bellman等式改进策略的方式为：**将策略选择的动作改为当前最优的动作**，而不是像之前那样对每种可能的动作进行求和。易知：选择当前最优动作相当于将所有的概率都赋给累积奖赏值最大的动作，因此每次改进都会使得值函数单调递增。

$$\pi'(x) = \arg \max_{a \in A} Q^\pi(x, a)$$

将策略评估与策略改进结合起来，我们便得到了生成最优策略的方法：先给定一个随机策略，现对该策略进行评估，然后再改进，接着再评估/改进一直到策略收敛、不再发生改变。这便是策略迭代算法，算法流程如下所示：

输入: MDP 四元组 $E = \langle X, A, P, R \rangle$;
累积奖赏参数 T .

过程:

```

1:  $\forall x \in X : V(x) = 0, \pi(x, a) = \frac{1}{|A(x)|}$ ; 随机策略
2: loop
3:   for  $t = 1, 2, \dots$  do
4:      $\forall x \in X : V'(x) = \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{t} R_{x \rightarrow x'}^a + \frac{t-1}{t} V(x) \right)$ ; 使用动态规划法计算
5:     if  $t = T + 1$  then
6:       break
7:     else
8:        $V = V'$ 
9:     end if
10:  end for 得到当前策略的T步状态值函数
11:  $\forall x \in X : \pi'(x) = \arg \max_{a \in A} Q(x, a)$ ; 依据状态值函数更新策略
12: if  $\forall x : \pi'(x) = \pi(x)$  then
13:   break
14: else
15:    $\pi = \pi'$ 
16: end if
17: end loop
```

输出: 最优策略 π

<http://blog.csdn.net/u011826404>

可以看出：策略迭代法在每次改进策略后都要对策略进行重新评估，因此比较耗时。若从最优化值函数的角度出发，即先迭代得到最优的值函数，再来计算如何改变策略，这便是值迭代算法，算法流程如下所示：

输入：MDP 四元组 $E = \langle X, A, P, R \rangle$;

累积奖赏参数 T ;

收敛阈值 θ .

过程:

1: $\forall x \in X : V(x) = 0$;

2: **for** $t = 1, 2, \dots$ **do**

3: $\forall x \in X : V'(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a (\frac{1}{t} R_{x \rightarrow x'}^a + \frac{t-1}{t} V(x'))$; **每次都选择最优动作**

4: **if** $\max_{x \in X} |V(x) - V'(x)| < \theta$ **then**


5: **break**

6: **else**

7: $V = V'$

8: **end if**

9: **end for**  **得到了最优的值函数**

输出：策略 $\pi(x) = \arg \max_{a \in A} Q(x, a)$  **根据最优值函数来改变策略**

<http://blog.csdn.net/u011826404>

16.4 蒙特卡罗强化学习

在现实的强化学习任务中，环境的转移函数与奖赏函数往往很难得知，因此我们需要考虑在不依赖于环境参数的条件下建立强化学习模型，这便是**免模型学习**。蒙特卡罗强化学习便是其中的一种经典方法。

由于模型参数未知，状态值函数不能像之前那样进行全概率展开，从而运用动态规划法求解。一种直接的方法便是通过采样来对策略进行评估/估算其值函数，**蒙特卡罗强化学习正是基于采样来估计状态-动作值函数**：对采样轨迹中的每一对状态-动作，记录其后的奖赏值之和，作为该状态-动作的一次累积奖赏，通过多次采样后，使用累积奖赏的平均作为状态-动作值的估计，并引入 **ϵ -贪心策略**保证采样的多样性。

输入: 环境 E ;
 动作空间 A ;
 起始状态 x_0 ;
 策略执行步数 T .

过程:

```

1:  $Q(x, a) = 0$ ,  $\text{count}(x, a) = 0$ ,  $\pi(x, a) = \frac{1}{|A(x)|}$ ; 随机策略
2: for  $s = 1, 2, \dots$  do
3:   在  $E$  中执行策略  $\pi$  产生轨迹
      $\langle x_0, a_0, r_1, x_1, a_1, r_2, \dots, x_{T-1}, a_{T-1}, r_T, x_T \rangle$ ;
4:   for  $t = 0, 1, \dots, T-1$  do
5:      $R = \frac{1}{T-t} \sum_{i=t+1}^T r_i$ ; 状态-动作的T-t步累积奖赏
6:      $Q(x_t, a_t) = \frac{Q(x_t, a_t) \times \text{count}(x_t, a_t) + R}{\text{count}(x_t, a_t) + 1}$ ; 更新状态-动作值函数
7:      $\text{count}(x_t, a_t) = \text{count}(x_t, a_t) + 1$ 
8:   end for
9:   对所有已见状态  $x$ :
     
$$\pi(x, a) = \begin{cases} \arg \max_{a'} Q(x, a'), & \text{以概率 } 1 - \epsilon; \\ \text{以均匀概率从 } A \text{ 中选取动作,} & \text{以概率 } \epsilon. \end{cases}$$

10: end for
输出: 策略  $\pi$ 

```

<http://blog.csdn.net/u011826404>

在上面的算法流程中, 被评估和被改进的都是同一个策略, 因此称为同策略蒙特卡罗强化学习算法。引入 ϵ -贪心仅是为了便于采样评估, 而在使用策略时并不需要 ϵ -贪心, 那能否仅在评估时使用 ϵ -贪心策略, 而在改进时使用原始策略呢? 这便是异策略蒙特卡罗强化学习算法。

输入: 环境 E ;

动作空间 A ;

起始状态 x_0 ;

策略执行步数 T .

过程:

- 1: $Q(x, a) = 0$, $\text{count}(x, a) = 0$, $\pi(x, a) = \frac{1}{|A(x)|}$;
 - 2: **for** $s = 1, 2, \dots$ **do**
 - 3: 在 E 中执行 π 的 ϵ -贪心策略产生轨迹
 $\langle x_0, a_0, r_1, x_1, a_1, r_2, \dots, x_{T-1}, a_{T-1}, r_T, x_T \rangle$;
 - 4: $p_i = \begin{cases} 1 - \epsilon + \epsilon/|A|, & a_i = \pi(x); \text{选择当前最优动作的概率} \\ \epsilon/|A|, & a_i \neq \pi(x), \text{选择随机动作的概率} \end{cases}$
 - 5: **for** $t = 0, 1, \dots, T-1$ **do**
 - 6: $R = \frac{1}{T-t} \sum_{i=t+1}^T (r_i \times \prod_{j=i}^{T-1} \frac{1}{p_j})$; 修正的累积奖赏
 - 7: $Q(x_t, a_t) = \frac{Q(x_t, a_t) \times \text{count}(x_t, a_t) + R}{\text{count}(x_t, a_t) + 1}$;
 - 8: $\text{count}(x_t, a_t) = \text{count}(x_t, a_t) + 1$
 - 9: **end for**
 - 10: $\pi(x) = \arg \max_{a'} Q(x, a')$ → 根据估计的状态-动作值函数改变策略
 - 11: **end for**
- 输出: 策略 π

<http://blog.csdn.net/u011826404>

16.5 ALPHAGO原理浅析

本篇一开始便提到强化学习是AlphaGo的核心技术之一，刚好借着这个东风将AlphaGo的工作原理了解一番。正如人类下棋那般“手下一步棋，心想三步棋”，Alphago也正是这个思想，当处于一个状态时，机器会暗地里进行多次的尝试/采样，并基于反馈回来的结果信息改进估值函数，从而最终通过增强版的估值函数来选择最优的落子动作。

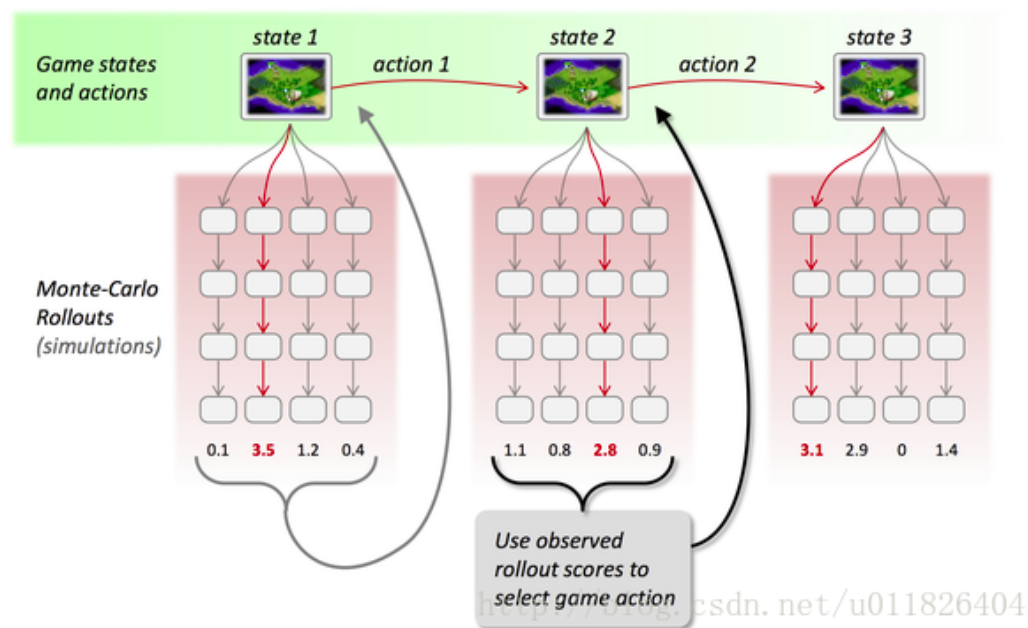
其中便涉及到了三个主要的问题：（1）如何确定估值函数（2）如何进行采样（3）如何基于反馈信息改进估值函数，这正对应着AlphaGo的三大核心模块：深度学习、蒙特卡罗搜索树、强化学习。

1.深度学习（拟合估值函数）

由于围棋的状态空间巨大，像蒙特卡罗强化学习那样通过采样来确定值函数就行不通了。在围棋中，**状态值函数**可以看作是一种局面函数，**状态-动作值函数**可以看作一种策略函数，若我们能获得这两个估值函数，便可以根据这两个函数来完成：(1)衡量当前局面的价值；(2)选择当前最优的动作。那如何精确地估计这两个估值函数呢？这就用到了深度学习，通过大量的对弈数据自动学习出特征，从而拟合出估值函数。

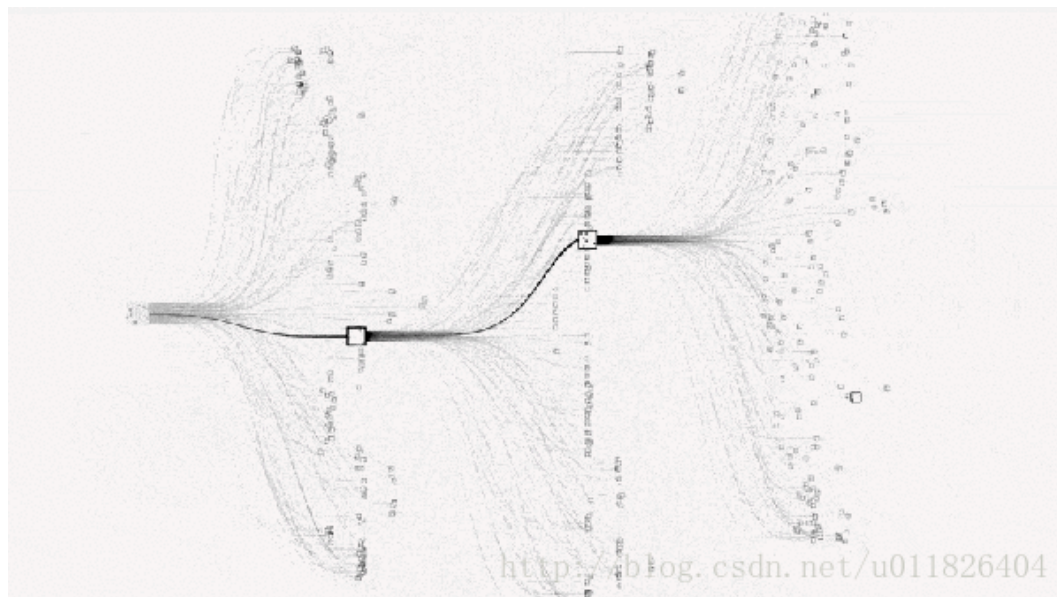
2.蒙特卡罗搜索树（采样）

蒙特卡罗树是一种经典的搜索框架，它通过反复地采样模拟对局来探索状态空间。具体表现在：从当前状态开始，利用策略函数尽可能选择当前最优的动作，同时也引入随机性来减小估值错误带来的负面影响，从而模拟棋局运行，使得棋盘达到终局或一定步数后停止。



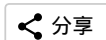
3.强化学习（调整估值函数）

在使用蒙特卡罗搜索树进行多次采样后，每次采样都会反馈后续的局面信息（利用局面函数进行评价），根据反馈回来的结果信息自动调整两个估值函数的参数，这便是强化学习的核心思想，最后基于改进后的策略函数选择出当前最优的落子动作。



在此，强化学习就介绍完毕~同时也意味着大口小口地啃完了这个西瓜，十分记得去年双11之后立下这个Flag，现在回想起来，大半年的时间里在嚼瓜上还是花费了不少功夫。有人说：当你阐述的能让别人看懂才算是真的理解，有人说：在写的过程中能发现那些只看书发现不了的东西，自己最初的想法十分简单：当健忘症发作的时候，如果能看到之前按照自己思路写下的文字，回忆便会汹涌澎湃一些~

最后，感谢自己这大半年以来的坚持~Get busy living, or get busy dying!



[置顶] 机器学习该如何入门
([http://spingspark.com/java...
%e6%9c%ba%e5%99%a8%e...](http://spingspark.com/java...%e6%9c%ba%e5%99%a8%e...))
七月 25, 2017
在“java”中

程序员们，AI来了，机会来了，
危机也来了
(<http://spingspark.com/java...>)
七月 11, 2017
在“java”中

转 程序员带你一步步分析AI如何
玩Flappy Bird Yao--靠自己
([http://spingspark.com/java...
%e7%a8%8b%e5%ba%8f%e5...](http://spingspark.com/java...%e7%a8%8b%e5%ba%8f%e5...))
bird-yao-
%e9%9d%a0%e8%87%aa%e...
七月 12, 2017
在“java”中

← 原 Javascript之同步与异步编程 沉非 (<http://spingspark.com/uncategorized/%e5%8e%9f-javascript%e4%b9%8b%e5%90%8c%e6%ad%a5%e4%b8%8e%e5%bc%82%e6%ad%a5%e7%bc%96%e7%a8%8b-%e6%b2%89%e9%9d%9e/>)

原 荐 数据库中间件 MyCAT源码分析 —— 跨库两表Join 芋艿V 开源马克杯 领取时间：2014-01-14 开源马克杯是开源中国定制的“高大上”Coders 喝水利器！ 领取条件：购买或拥有开源马克杯的OSCer可领取 → (<http://spingspark.com/java/%e5%8e%9f-%e8%8d%90-%e6%95%b0%e6%8d%ae%e5%ba%93%e4%b8%ad%e9%97%b4%e4%bb%b6-mycat%e6%ba%90%e7%a0%81%e5%88%86%e6%9e%90-%e8%b7%a8%e5%ba%93%e4%b8%a4%e8%a1%a8join-%e8%8a%8b%e8%89%bfv-285/>)

近期文章

» wolfx / vue-components-searcher (<http://spingspark.com/java/wolfx-vue-components-searcher/>)

» 原 php设备检测工具类 zhiqiangw (<http://spingspark.com/java/%e5%8e%9f-php%e8%ae%be%e5%a4%87%e6%a3%80%e6%b5%8b%e5%b7%a5%e5%85%b7%e7%b1%bb-zhiqiangw-2/>)

» echarts添加点击事件
(<http://spingspark.com/uncategorized/echarts%e6%b7%bb%e5%8a%a0%e7%82%b9%e5%87%bb%e4>

» 原 每个前端开发者必会的20个JavaScript面试题 前端攻城狮-Dawn
(<http://spingspark.com/%e5%89%8d%e7%ab%af/%e5%8e%9f-%e6%af%8f%e4%b8%aa%e5%89%8d%e7%ab%af%e5%bc%80%e5%8f%91%e8%80%85%e5%bf%85%e4%e5%89%8d%e7%ab%af%e6%94%bb%e5%9f%8e%e7%8b%ae-dawn/>)