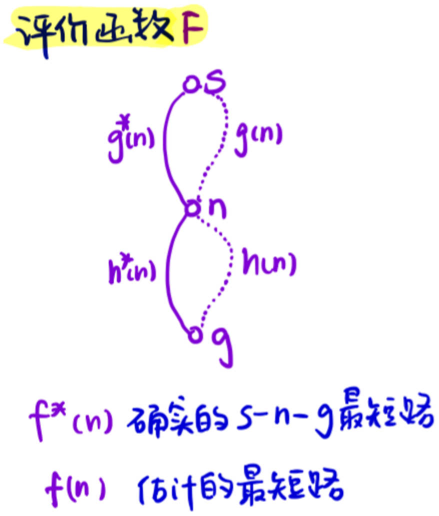
人智导期末复习

一、图搜索

评价函数



A算法

```
open, closed = {src}, {}
while True:
    if open.empty():
        return None

    取出 n := open 中首个节点，将之加入 closed
    if n == goal:
        return n

    扩展 n 的子节点：
        if mj not in open and mj not in closed:
            mj.parent = n
            f(mj) = g(n, mj) + h(mj)

        elif mk in open and f(n, mk) < f(mk):
            mk.parent = n
            f(mk) = f(n, mk)

        elif ml in closed and f(n, ml) < f(ml):
            ml.parent = n
            f(m_l) = f(n,m_l)
            将 ml 插入 open
    按照 f 升序排序 open
```

A*算法

条件：$h(n) \le h^*(n)$

结论：

- 若初始节点到目标节点间存在路径，A* 必能找到最佳解；
- 若 $h_2(n) > h_1(n)$, 则 A1 扩展的节点数 ≥ A2 扩展的节点数。

  注意: 若$h_2(n) \geq h_1(n)$, 则当$h_2(n) = h_1(n)$时，由于具有相同 f 值的点被扩展顺序的随机性，不能保证结论成立。

h 单调性

对节点 n_i, n_j, n_j是n_i的子节点，满足

$$\begin{cases} h(n_i) - h(n_j) \leq c(n_i, n_j) \\ h(t) = 0 \end{cases}$$

称 h 具有单调性。

- 若 h 单调，A* 条件必然满足。
- 若 h 单调，则A*扩展了节点 n 后，就已经找到了到达节点 n 的最佳路径。 即 $g(n) = g^*(n)$

改进A*算法

根据以下结论：

- OPEN 表中任一满足 $f(n) < f^*(s)$ 的节点定会被扩展。
- 被扩展的节点定满足 $f(n) \leq f^*(s)$。
- 当 $h(n) \equiv 0$ 时，h 单调。

```
# ...
fm = 0
while True:
    if open.empty():
        return None
    Nest = {n for n in open if f(n) < fm}
    if not Nest.empty():
        n = NEST 中 g 最小的节点
    else:
        取出 n := open 中首个节点
    将 n 加入 closed
    # ...
```

Viterbi算法

$$P(T_{i,j}) = \begin{cases} \max\limits_k \{P(T_{i-1,k})P(w_{i,j}|T_{i-1,k})\} & i \geq 2 \\ P(w_{i,j}) & i = 1 \end{cases}$$

二、对抗搜索

$\alpha - \beta$ 搜索

该算法和极小化极大算法所得结论相同，但剪去了不影响最终决定的分枝。

```
def alphabeta(node, α=-∞, β=∞, maximizingPlayer):
    if node is 叶节点:
        return 节点值
    if maximizingPlayer:
        v = -∞
        for child in node.children:
```

```
            v = max(v, alphabeta(child, α, β, FALSE))
            α = max(α, v)
            if β <= α:
                break # β裁剪
        return v
    else:
        v = ∞
        for child in node.children:
            v = min(v, alphabeta(child, α, β, TRUE))
            β = min(β, v)
            if β <= α:
                break # α裁剪
        return v
```

MCTS

不依赖于专家知识。

```
def UCTSEARCH(s0):
  以状态s0创建根节点v0;
  while 尚未用完计算时长 do:
    vl = TREEPOLICY(v0);  # 获得待扩展节点
    Δ = DEFAULTPOLICY(s(vl ));  # 随机模拟至终局，获得收益
    BACKUP(vl , Δ);  # 回传收益，逐层取反
  return a(BESTCHILD(v0, 0));

def TREEPOLICY(v):
    ''' 只要一个节点没有孙子节点，就表明它可能可扩展 '''
  while 节点v不是终止节点 do:
    if 节点v是可扩展的 then:
      return EXPAND(v)
    else:
      v = BESTCHILD(v, c)
  return v

def BESTCHILD(v, c):
    '''
    对于MCTS, I(v') = Q(v')
    对于UCT, I(v')= Q(v') + sqrt(2ln(N(v))/N(v'))
    '''
  return argmax v'∈children of v: I(v')
```

三、高级搜索 (组合优化)

模拟退火输出的是最后一个解，遗传算法输出历代中的最优解。

局部搜索算法

思想：在邻域中有方向地搜索。

防止陷入局部最优：

- 多次随机选择初始点
- 从邻域内随机选点，选取的概率与指标函数的大小正相关

变步长，为了收敛。

模拟退火算法

```
t = Tmax
随机选择解i，计算f(i)
while not 满足终止条件:
    if t 温度下的迭代结束:
        t = Drop(t)
        continue

    从 i 的邻域中随机选解 j
    if f(j) < f(i) or Random(0, 1) < e^(-Δf/t):
        i, f(i) = j, f(j)
return i
```

- 起始温度的选择
- 特定温度 t 下中止迭代的条件
  - 固定迭代次数
  - 固定接受状态数
- 温度下降方法 `Drop`
  - 等比例
- 算法终止条件
  - 对温度设置阈值
  - 固定温度下降次数
  - 如果相邻的n个温度中，所得解的指标函数值无变化，则中止算法

遗传算法

```
# 输入: 种群规模 N, 交叉概率 pc, 变异概率 pm
随机生成 N 个染色体作为初始群体 X
while True:
    对于群体中每个染色体计算其适应值
    if 满足停止条件:
        break

    根据适应值计算选择概率，在种群中随机选择 N 个染色体
    依概率 pc 进行交叉
    依概率 pm 发生变异

    用新种群替代旧种群
return 进化历程中的最优解
```

必须**记录进化历程中的最优解**，才能以概率1收敛到最优解。

选择

- 模拟轮盘赌
- 确定性法

“确定性”法

对于规模为N的群体，一个选择概率为$p(x_i)$的染色体x_i被选择次数的期望值$e(x_i)$：

$$e(x_i) = p(x_i)N$$

对于群体中的每一个x_i，首先选择 $\lfloor e(x_i) \rfloor$ 次。这样共得到 $\sum_{i=1}^{N}\lfloor e(x_i) \rfloor$ 个染色体。然后按照$e(x_i)-\lfloor e(x_i) \rfloor$从大到小对染色体排序，依次取出 $N-\sum_{i=1}^{N}\lfloor e(x_i) \rfloor$ 个染色体，这样就得到了N个染色体。

关于适应函数加速、平均、交叉和变异的方法等其它细节，见遗传算法.

四、机器学习

朴素贝叶斯

$$\begin{aligned} y &= arg\max_{c_k} P(Y=c_k)P(X=x|Y=c_k) \\ &= arg\max_{c_k} P(Y=c_k)\prod_{i=1}^{n} P(X^{(i)}=x^{(i)}|Y=c_k) \end{aligned}$$

参数估计的平滑

$$\begin{aligned} P(X^{(j)}=a_{ij}|Y=c_k) &= \frac{\sum_{i=1}^{N} I(x_i^{(j)}=a_{jl}, y_i=c_k)+\lambda}{\sum_{i=1}^{N} I(y_i=c_k)+S_j\lambda} \\ P(Y=c_k) &= \frac{\sum_{i=1}^{N} I(y_i=c_k)+\lambda}{N+K\lambda} \end{aligned}$$

其中 K 是标签数，S_j 是特征 $X^{(j)}$ 可能的取值个数.

$\lambda=1$时，称 Laplace 平滑。

SVM

线性可分

$$\min_{w,b} \frac{1}{2}||w||^2, \text{s.t. } y(w^T x_i + b) \geq 1$$

拉格朗日对偶问题：

$$\begin{aligned} \min_{\alpha} \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{m}\alpha_i \\ \text{s.t. } \sum_{i=1}^{m}\alpha_i y_i = 0, \alpha_i \geq 0 \end{aligned}$$

其中$w=\sum_{i=1}^{m}\alpha_i y_i x_i$,

对于支持向量x_j ($\alpha_j>0$), $b=y_j-w^T x_j$,

附：根据KKT条件

$$\alpha_i[y(w^T x_i + b) - 1] = 0$$

$\alpha > 0$的点满足 $y(w^T x_i + b) = 1$ 构成支持向量。

线性不可分

$$\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} \xi_i,$$
$$\text{s.t. } y(w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

对偶问题:

拉格朗日对偶问题:

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{m} \alpha_i$$
$$\text{s.t. } \sum_{i=1}^{m} \alpha_i y_i = 0, C \geq \alpha_i \geq 0$$

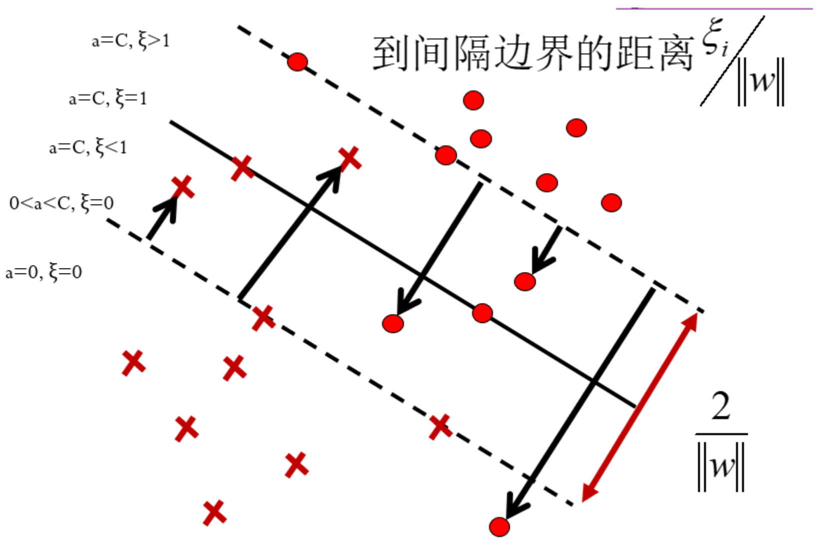
其中$w = \sum_{i=1}^{m} \alpha_i y_i x_i$,

对于支持向量x_j ($C \geq \alpha_j > 0$), $b = y_j - w^T x_j$.

ξ实际上是 hinge loss: $\xi = \max(0, 1 - y(w^T x + b))$

根据KKT条件:

$$\alpha_i[y(w^T x_i + b) - 1 - \xi_i] = 0$$
$$\alpha_i + \mu_i = C$$
$$\mu_i \xi_i = 0$$



非线性

将上述$x_i^T x_j$换成$K(x_j, x_j)$.

对于支持向量x_j, $b = y_j - \sum \alpha_i y_i K(x_i, x_j)$

$f(x) = sign(\sum \alpha_i y_i K(x_i, x) + b)$

决策树

信息增益

$$
H(D) = -\sum \frac{C_k}{D} \log \frac{C_k}{D}
$$
$$
H(D|A) = -\sum \frac{D_i}{D} \sum \frac{D_{ik}}{D_i} \log \frac{D_{ik}}{D_i}
$$
$$
g(D,A) = H(D) - H(D|A)
$$

ID3算法

1. 递归终止条件：样本为同一类，或可用特征为空
2. 计算特征集A中各特征信息增益，选择最大者A_g
3. 若 $g(D, A_g) < \epsilon$, 递归中止
4. 依 A_g将训练集D划分为若干子集，作为D的子节点。若某子集为空，则递归中止
5. 对子节点调用递归算法。

缺点：依据信息增益决策，倾向于选择分枝多的属性

信息增益比

$$
H_A(D) = -\sum \frac{D_i}{D} \log \frac{D_i}{D}
$$
$$
g_R(D,A) = g(D,A)/H_A(D)
$$

缺点：倾向于选择分割不均匀的属性。

C4.5算法

将 ID3 算法的第 2 步改为：先选择n个信息增益大的特征，再从这n个特征中选择信息增益比最大的特征。

后剪枝

先生成整棵树，再自下而上地剪枝。

- 基于验证集性能
- 基于损失函数

$$
C_a(T) = \sum_{t=1}^{|T|} N_t H_t(T) + \alpha |T|
$$
$$
H_t(T) = -\sum \frac{N_{tk}}{N_t} \log \frac{N_{tk}}{N_t}
$$

随机森林

多决策树投票 + 随机采样生成

五、深度学习

激活函数

$$
\sigma(x) = \frac{1}{1+e^{-x}}
$$
$$
\sigma'(x) = \sigma(x)(1-\sigma(x))
$$
$$
\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}
$$
$$
softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}
$$

损失函数

$$L_{entropy} = -\sum_i y_i \log p_i$$

$$L_2 = \frac{1}{2}||p-y||_2^2$$

卷积

输入$n \times n$, 卷积核 $f \times f$, 步长为d, 填充$p \times p$

卷积后输出各维度形状:

$$[\frac{n+2p-f}{s}]+1$$

序列模型

word2vec

基本思想：相邻的词有相近的语义。

CBOW：用 context 预测 center

Skip gram: 用 center 预测 context

加速 softmax: 用霍夫曼树将 softmax 优化为 $O(\log N)$ 次 sigmoid.