1. 解：

首先$\beta_{A,i}$一定非负，这样才符合市场含义。

方法 1：
根据

$$r_A(t) = \sum_{i\in sector(A)} \beta_{A,i} r_i(t) + \epsilon_A(t)$$

得

$$\epsilon_A(t) = r_A(t) - \sum_{i\in sector(A)} \beta_{A,i} r_i(t)$$

显然，对于一个良好的对ϵ_A的估计，$\sum_t \left(\epsilon_A(t)\right)^2$应当尽可能小，这样才能尽可能排除掉行业回报。
故转换成凸优化问题：

$$\min \sum_t \left(\epsilon_A(t)\right)^2$$

$$s.t. \sum_{i\in sector(A)} \beta_{A,i} = 1, \beta_{A,i} \geq 0$$

这样，就可以使用 python 包 cvxpy 进行凸优化求解。代码运行结果见 test.ipynb

方法 2：
不妨记所有股票的集合为E，总天数为D，假设$\epsilon_A(t) \sim N(\mu, \sigma^2)$，则可以用 EM 算法
先初始化$\beta_{A,i} = \frac{1}{|sector(A)|}, \forall i \in sector(A)$，则有

E-Step:

$$\mu = \frac{1}{D|E|} \sum_{A\in E} \sum_t \left(r_A(t) - \sum_{i\in sector(A)} \beta_{A,i} r_i(t) \right)$$

$$\sigma^2 = \frac{1}{D|E|} \sum_{A\in E} \sum_t \left(r_A(t) - \sum_{i\in sector(A)} \beta_{A,i} r_i(t) - \mu \right)^2$$

M-Step:

$$\log P = -\frac{D|E|}{2} \log(\sigma^2) - \frac{1}{2} \sum_{A\in E} \sum_t \frac{\left(r_A(t) - \sum_{i\in sector(A)} \beta_{A,i} r_i(t) - \mu\right)^2}{\sigma^2}$$

$$\max(\log P)$$

$$s.t. \sum_{i\in sector(A)} \beta_{A,i} = 1$$

理论上可解，但经过试验，发现 cvxpy 无法解出符合约束条件的解，且时间复杂度太高，故放弃这个方法。

2. 解：

当行业回报$r_i(t)$未知时，如果仍用方法一，则此时优化的目标函数变为非凸，无法使用，故不能使用方法一。
如果使用方法二，则理论上是可行的，但是时间复杂度太高，并且 cvxpy 无法计算出正确结果。

不妨记所有股票的集合为E，总天数为D，每个行业i所拥有的股票全集为$stock(i)$，则可以用 EM 算法

先初始化$\beta_{A,i} = \frac{1}{|sector(A)|}, \forall i \in sector(A)$，则有

E-step:

固定$\beta_{A,i}$，优化以下目标函数：

$$\min \sum_{A \in E} \sum_{t} \left(r_A(t) - \sum_{i \in sector(A)} \beta_{A,i} r_i(t) \right)^2$$

通过 cvxpy 求解，可以获得$r_i(t)$的估计值。

M-step:
固定$r_i(t)$，优化以下目标函数：

$$\min \sum_{t} \left(r_A(t) - \sum_{i \in sector(A)} \beta_{A,i} r_i(t) \right)^2$$

$$s.t. \sum_{i \in sector(A)} \beta_{A,i} = 1, \beta_{A,i} \geq 0$$

通过 cvxpy 求解，可以获得$\beta_{A,i}$的新估计值。

二者反复迭代，直到$\beta_{A,i}$的变化幅度很小为止。代码运行结果见 test.ipynb