Pythonで体験するベイズ推論 第5章

石田研究室　M1

西岡 知剛

損失関数

- **損失関数**：現在の推定値がどのくらい悪いのかを測る指標

$$L(\theta, \hat{\theta}) = f(\theta, \hat{\theta})$$

θは真のパラメータ, $\hat{\theta}$は推定値

- 有名な損失関数の例
  - **二乗誤差損失**

$$L(\theta, \hat{\theta}) = (\theta - \hat{\theta})^2$$

非対称な二乗誤差損失関数

$$L(\theta, \hat{\theta}) = \begin{cases} (\theta - \hat{\theta})^2 \ (\hat{\theta} < \theta) \\ c(\theta - \hat{\theta})^2 \ (\hat{\theta} \geq \theta) \text{ただし} 0 < c < 1 \end{cases}$$

(推定値が真値より大きい方が好ましいことを表している)

損失関数

- 有名な損失関数の例(続き)
  - **絶対損失**
    $$L(\theta, \hat{\theta}) = |\theta - \hat{\theta}|$$
  - **0-1損失**
    $$L(\theta, \hat{\theta}) = 1_{\hat{\theta} \neq \theta}$$
  - **対数損失**
    $$L(\theta, \hat{\theta}) = -\hat{\theta} \log(\theta) - (1 - \hat{\theta}) \log(1 - \theta), \hat{\theta} \in \{0,1\}, \theta \in \{0,1\}$$

実世界の損失関数

- 真のパラメータθを知ってたら推定する意味・・・
- 損失関数に実際に意味があるのは真のパラメータは未知であるという状況

- ベイズ推論では未知パラメータは事前/事後分布に従う確率変数と考える
  - 事後分布からのサンプルの値は、真の値が取る可能性のある値

  ➢その値から推定値に対する**期待損失**を計算できる
    - 期待損失は真の損失の推定値

ベイズ点推定

- 最終的には事後分布ではなくある1つの値(もしくはベクトル)に絞ることが必要
- この値をベイズ推論の事後分布から得ることが**ベイズ点推定**

- P(θ|X)をXを観測した後のθの事後分布とする
  - 以下は「θに対する推定値$\hat{\theta}$についての期待損失」と解釈できる

$$l(\hat{\theta}) = E_{\theta}[L(\theta, \hat{\theta})]$$

  - この値は、大数の法則を用いて近似できる(第4章)

$$\frac{1}{N}\sum_{i=1}^{N} L(\theta_i, \hat{\theta}) \approx E_{\theta}[L(\theta, \hat{\theta})] = l(\hat{\theta})$$

ベイズ点推定

- 期待値を介して損失を測るのは、MAP推定値よりも分布の情報を多く使っている(MAPは分布の最大値だけを求める)
  - 分布の情報を無視してしまうと、ロングテールな危険性に影響を受けすぎてしまい、自分がどれほどパラメータに無知であるかの情報も、推定値から失われてしまう

- 頻度主義の手法は、誤差を最小化することのみが目的で、推定誤差の与える結果の損失を考慮しない
- ベイズ点推定はもし推定値が間違いになりそうなら、結果の影響がマシな方に間違うようにしておける
  - 例：降水予報

例題：テレビ番組”The Price Is Right”の最適化

- ルール
  - 番組には２人の出場者がいる。
  - それぞれの出場者に、別々の賞品が二つずつ提示される。
  - それを見た後、出場者はそれぞれの二つの賞品の合計価格を予想する。
  - 予想した価格が実際の合計価格を上回った場合は、失格となる。
  - 予想価格と実際の合計価格との差が250ドル未満であれば、相手の分の賞品も獲得できる。

価格について確信が持てないのに、予想価格を合計価格を上回らないように近づけなければならない！！

例題：テレビ番組”The Price Is Right”の最適化

- 真の価格は正規分布に従うと仮定

$$\text{真の価格} \sim Normal(\mu_p, \sigma_p)$$

  - ここでは$\mu_p = 35{,}00,\ \sigma_p = 7{,}500$とする。

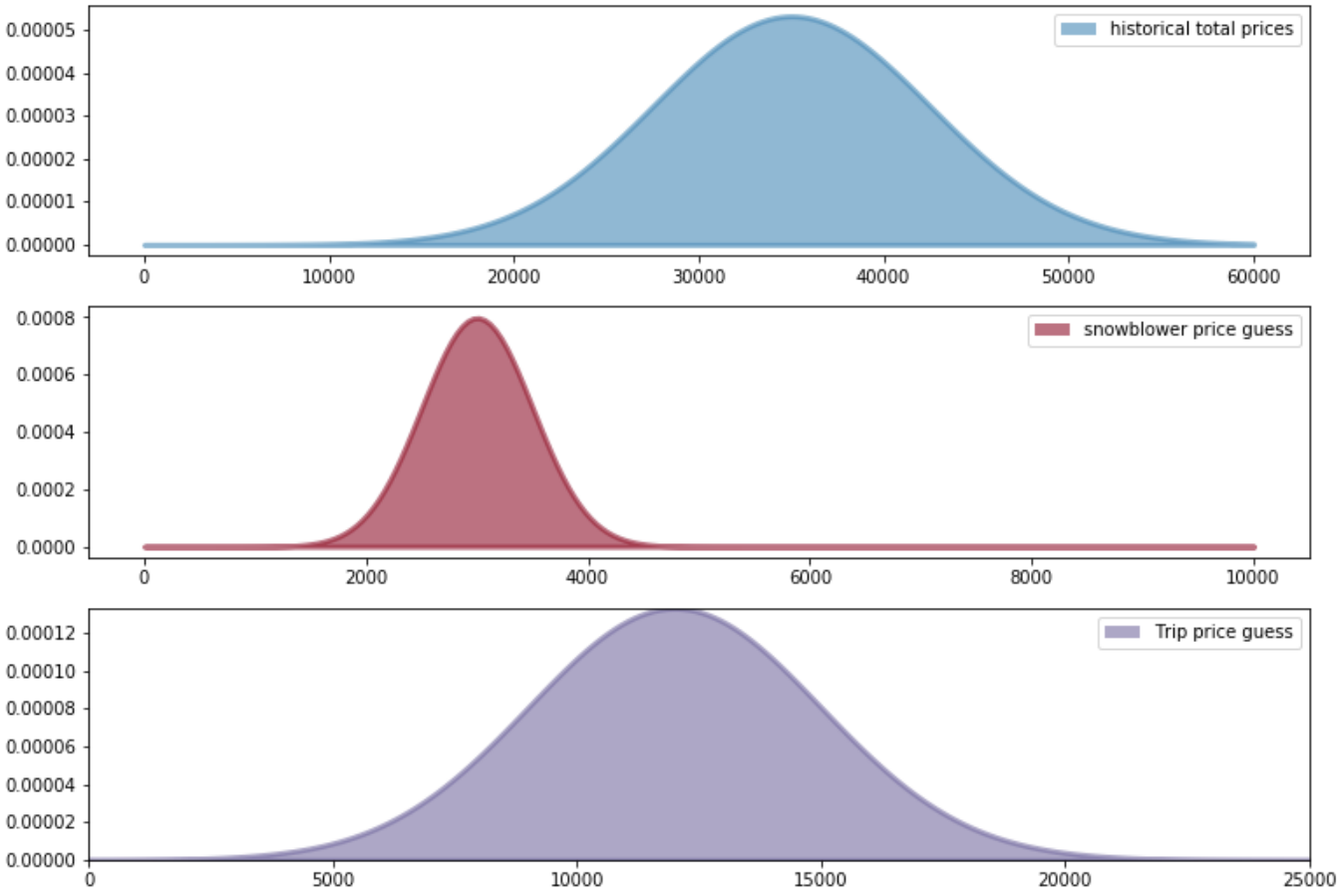
- それぞれの賞品価格についての信念も正規分布に従うと仮定

$$\text{賞品}_i \sim Normal(\mu_i, \sigma_i)$$

  - μ_iでどの程度の価格が正しいと思っているかを指定
  - σ_iでその予測がどのくらい確実と思っているかを表現

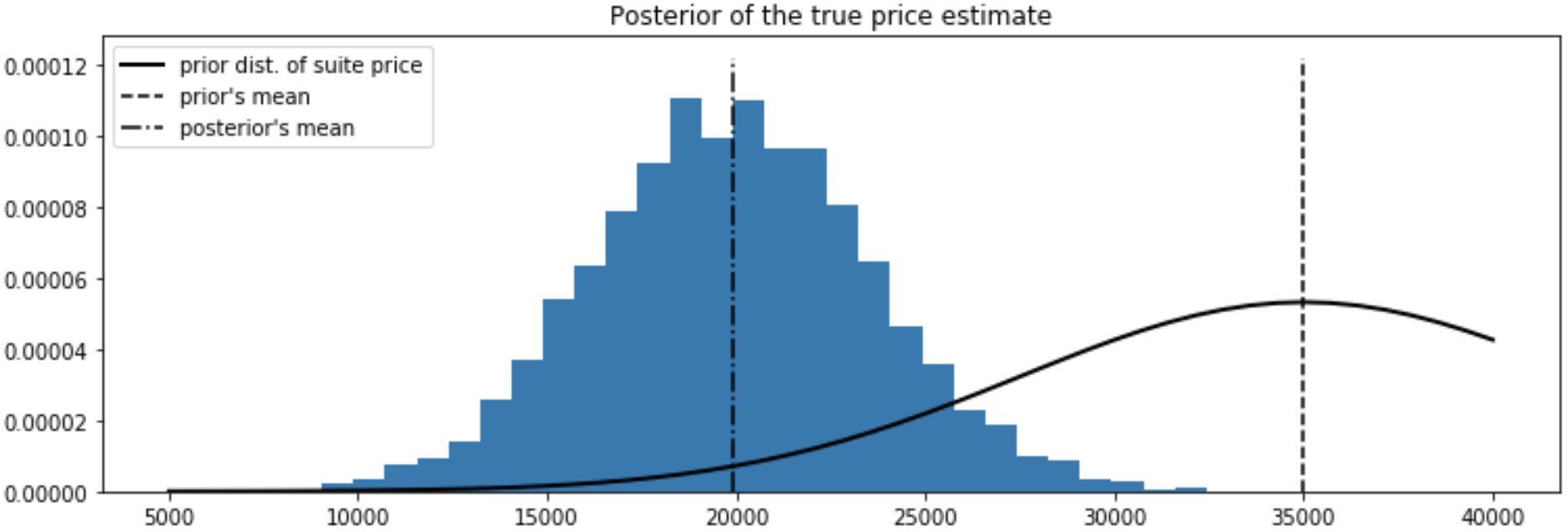
例題：テレビ番組”The Price Is Right”の最適化

トロント$\sim Normal(12000,3000)$
除雪機$\sim Normal(3000,500)$



例題：テレビ番組”The Price Is Right”の最適化

- 除雪機の価格、旅行の価格、予想合計価格(とその不確実さ)を考慮した結果、事前分布の平均価格より15,000ドル低い価格を推定



- 頻度主義では不確実にかかわらず35,000ドルと推定してしまう

例題：テレビ番組”The Price Is Right”の最適化

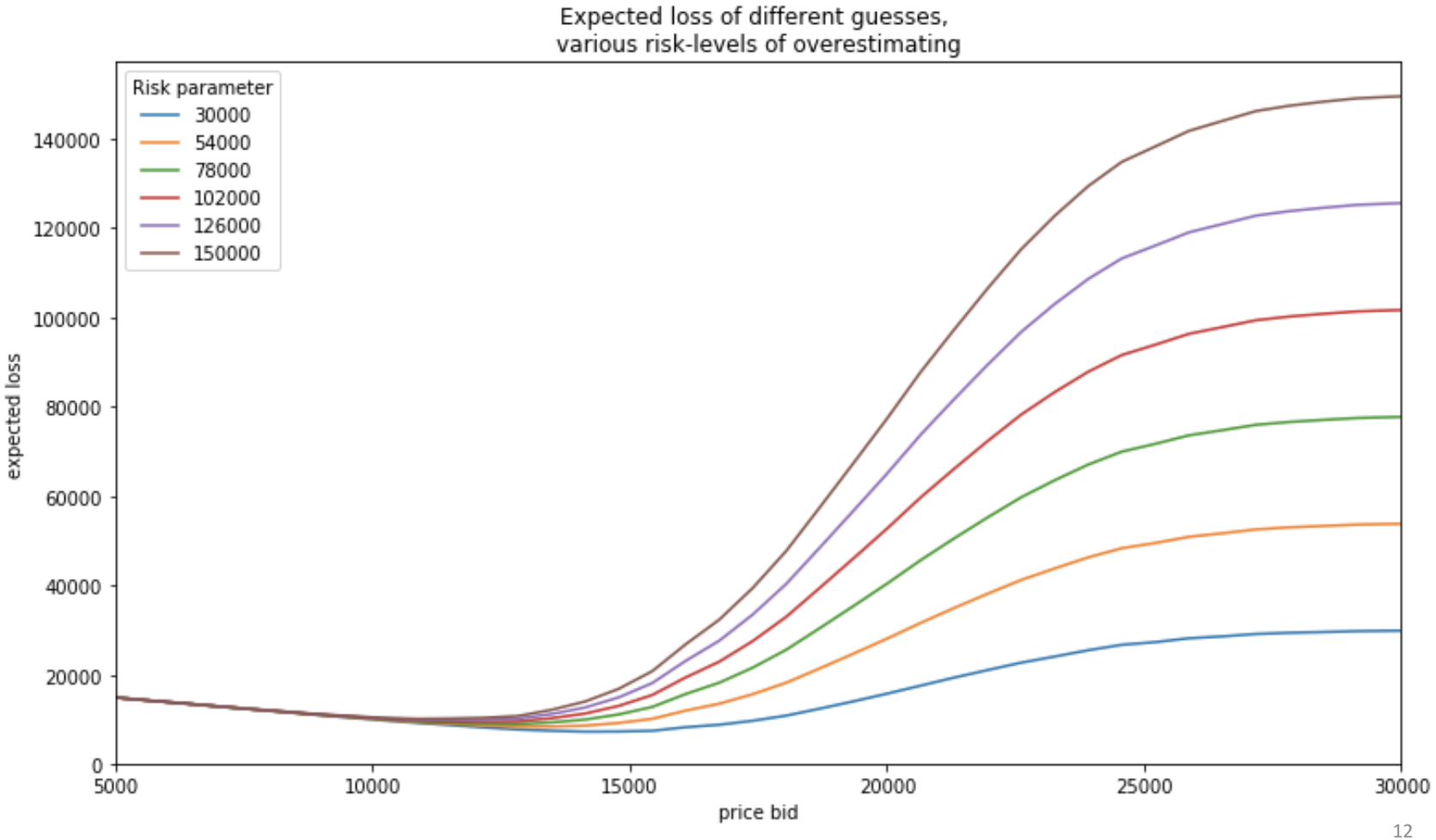
- 出場者の損失関数を定義してみる

```
def showdown_loss(guess, true_price, risk=80000):
    if true_price < guess:
        return risk
    elif abs(true_price - guess) <= 250:
        return  -2 * np.abs(true_price)
    else:
        return np.abs(true_price - guess - 250)
```

- riskは予想価格が真の価格を上回ったときに、どれだけ悪いかを表すパラメータ
  - riskを低くすると,予想が真値を上回ることをより許容する
- 予想の価格と真の価格の差が250ドル以下なら両方の賞品を獲得
  - このモデルでは真の価格の2倍を受け取る設定

例題：テレビ番組”The Price Is Right”の最適化

- riskパラメータを変化させると損失がどのように変化するか



例題：テレビ番組”The Price Is Right”の最適化

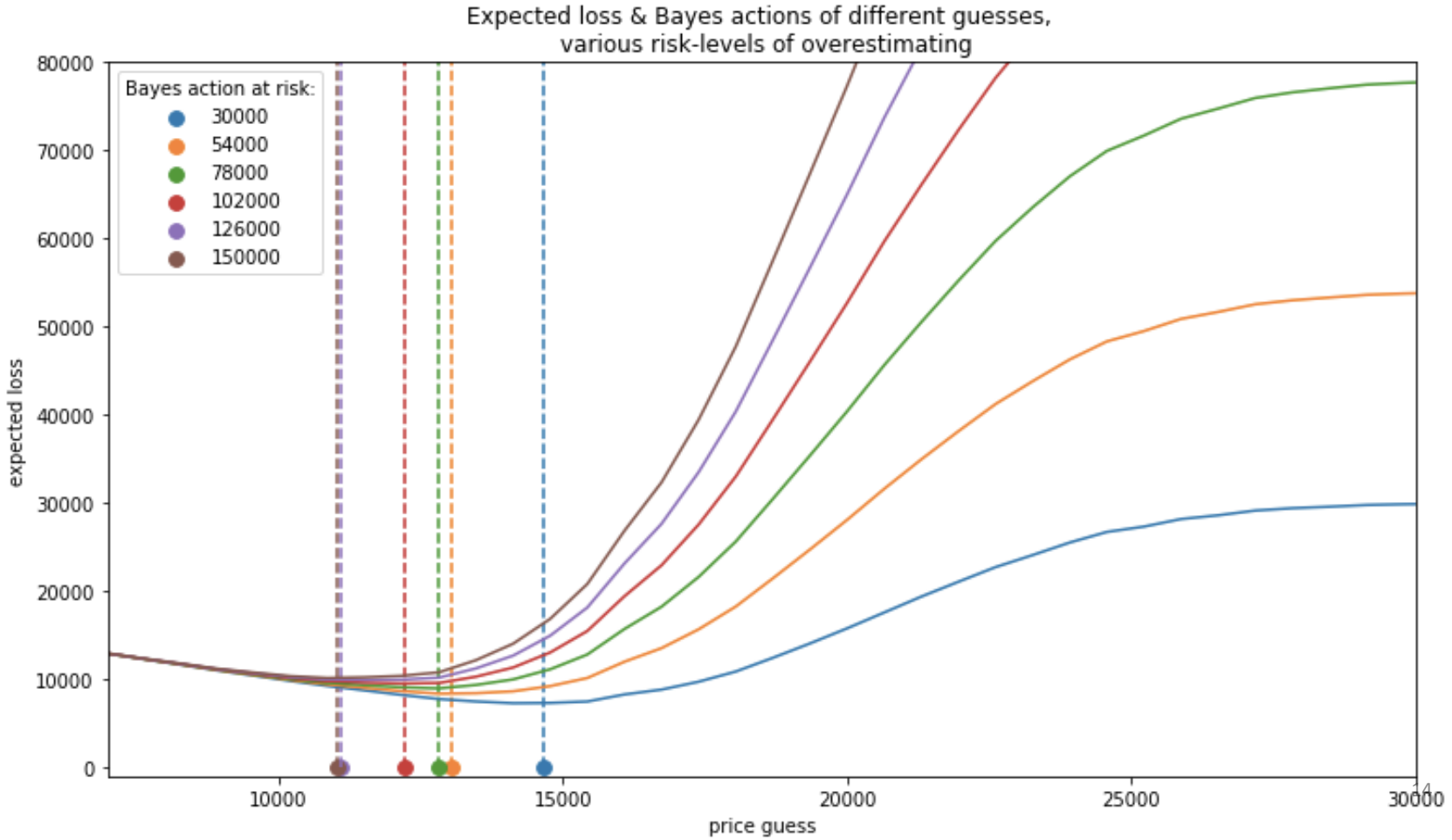
- 損失の最小化
  - 先程の図の各曲線の最小値を求める
  - 以下の期待損失を最小化

$$\arg\min_{\theta} E_{\theta}[L(\theta, \hat{\theta})]$$

- 期待損失を最小化するθは**ベイズ行動**と呼ばれる
  - Scipyの最適化モジュールで求めることができる
  - scipy.optimize.fminは1/多変数関数の極小値を求める

例題：テレビ番組”The Price Is Right”の最適化

- リスクを下げると、予想価格が上がる
- 事後平均(20,000)からは大きく予想価格が下がっている



近道

- いくつかの損失関数に対しては、そのベイズ行動が解析的に得られることが知られている
  - 二乗誤差…事後分布の平均
  - 期待絶対損失…事後分布の中央値
  - 0-1損失…MAP推定値

ベイズ手法を用いた機械学習

- 頻度主義の手法 — 最も精度の高いパラメータを求める
- 機械学習 — 最も正確な予測をするパラメータを求める
  - 予測の良さの指標と最小化する損失関数とが違っている事がよくある
  - 例えば、最小二乗線形回帰は最も単純な「学習」アルゴリズムだが、もし損失関数が二乗誤差損失でないなら、最適な予測結果は得られない
- ベイズ行動を求めることは、パラメータの精度ではなく、 任意の性能評価指標を最適化するパラメータを求めることに等しい

例題：株価の予測

- 株価の将来のリターンを0.01(1%)と仮定
- 損失をどのように評価するか？
  - 二乗誤差では符号が無視されてしまう
  - $\big(0.01-(-0.01)\big)^2=(0.01-0.03)^2=0.004$
  - -0.01と予測した場合投資しないため損をし、0.03と予想した場合得をする

  - もっと良い損失関数を！！(stock_loss関数)

```
def stock_loss(true_return, yhat, alpha=100.):
    if true_return * yhat < 0:
        # opposite signs, not good
        return alpha * yhat ** 2 - np.sign(true_return) * yhat \
            + abs(true_return)
    else:
        return abs(true_return - yhat)
```

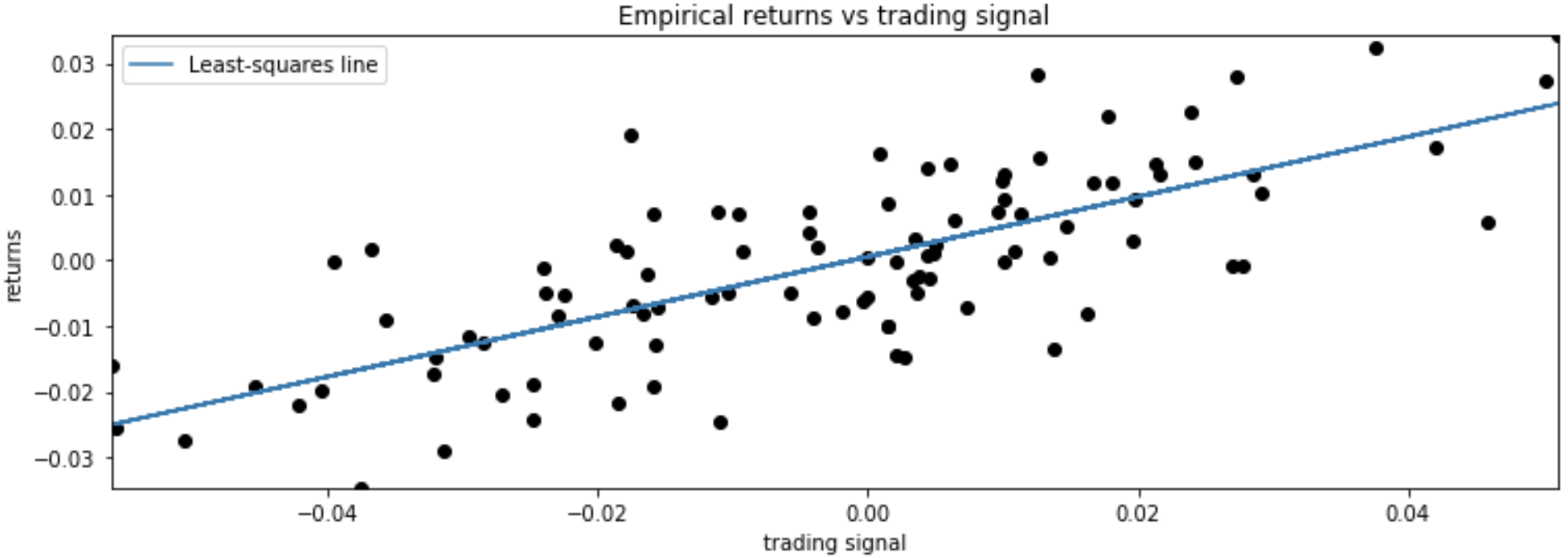
例題：株価の予測

- この損失は、符号は間違えたくないし、符号を間違えたまま値を大きくすることはもっとやりたくないということを反映

- 符号が正しいときはそれほど損失は増えない
  - 損失を0にしないのは、金融の世界では上振れリスク(正しい方向に大きく予測すること)も推奨されないため

例題：株価の予測

- 将来のリターンを良く予測していると思われる株式データ(trading signal)に対して回帰を実行してみる



なお、実際の金融データはこんな線形ではない模様

例題：株価の予測

- ここでは、単純なベイズ線形回帰をこのデータセットに適応
  - 以下のようなモデルを考える

$$R = \alpha + \beta x + \epsilon$$

  - α, βは未知のパラメータ, $\epsilon \sim Normal(0, 1/r)$
  - α, βに対する一般的な事前分布は正規分布
  - rについても, $\sigma = 1/\sqrt{r}$が0~100の一様分布を事前分布に設定 (r= $Uniform(0,100)^2$)

例題：株価の予測

- ある株式データxに対して、そのリターンは以下の分布を持つ

$$R_i(x) = \alpha_i + \beta_i x + \epsilon$$

  - $\epsilon \sim Normal(0, 1/r_i)$, iは事後サンプルのインデックス

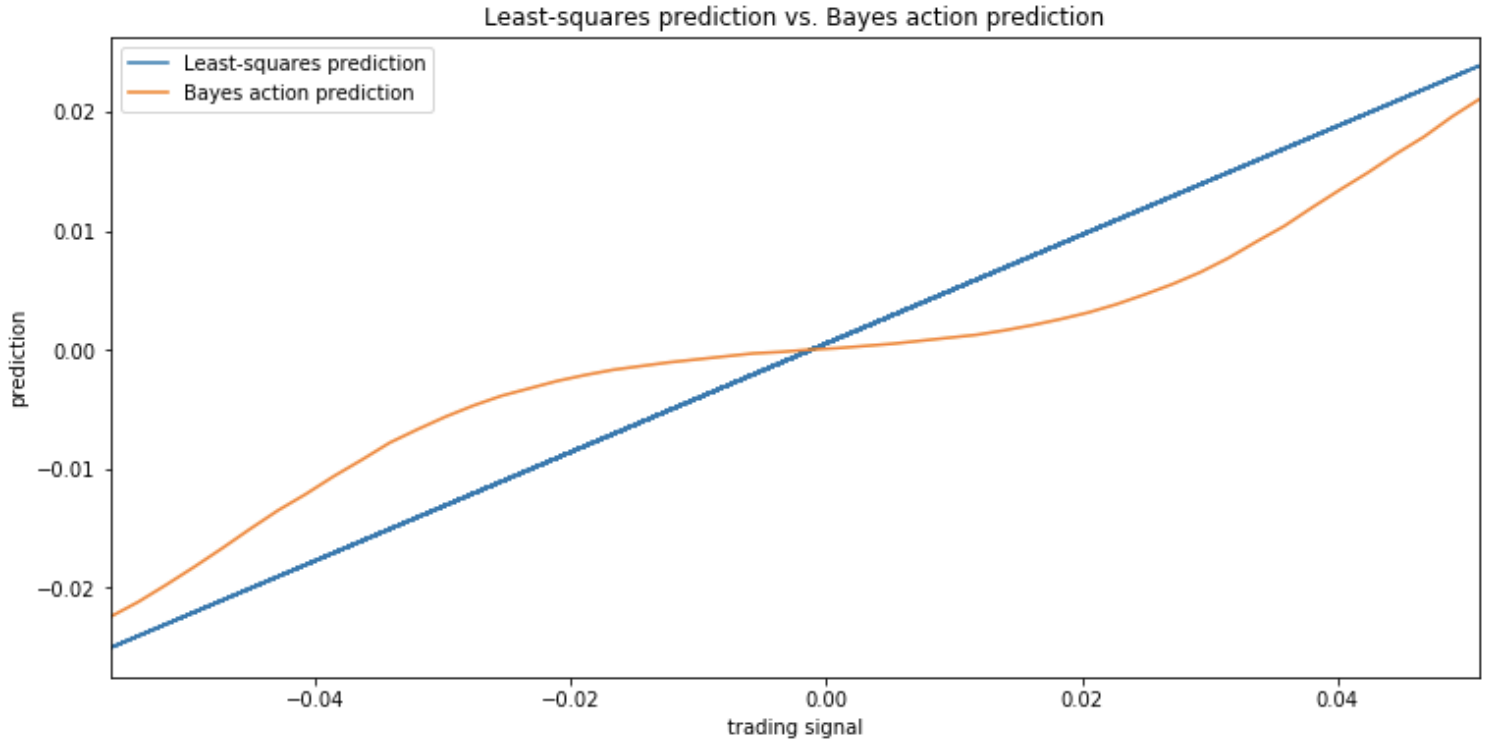
- 損失関数に対して、以下の問題の解を求める

$$\arg\min_r E_{R(x)}[L(R(x), r)]$$

  - このrが株式データxに対するベイズ行動

例題：株価の予測

- 株式データに対するベイズ行動のプロット



- 株式データが0に近い時、最も良い予測が0に近い値を取っている
  - なるべく取引しないで、確実と思うときだけ取引する！

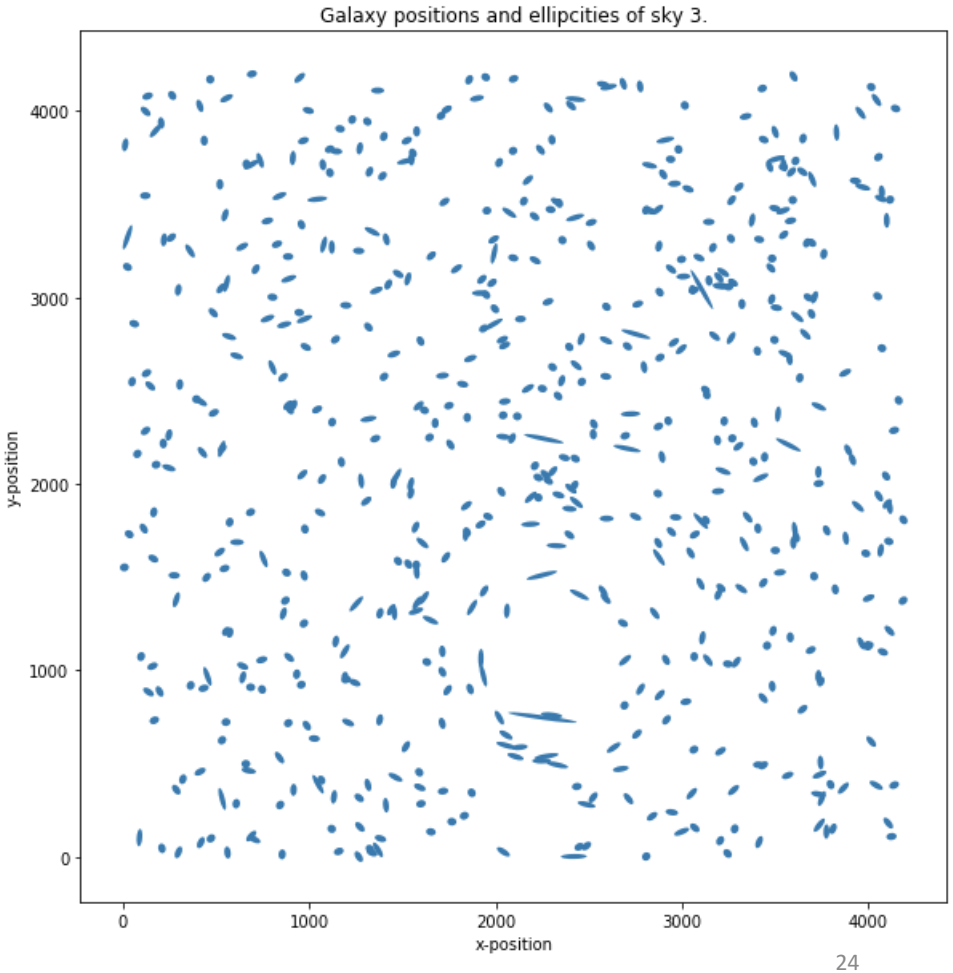
例題：Kaggleコンテスト「ダークマターの観測」

- コンテストの趣旨：ダークマターがどこに存在しているのかを予測
  - 優勝者Tim Salimansはベイズ推論を用いてハロー(ダークマターの塊)の場所を求めた

1. ハローの位置の事前分布p(x)を構築.
2. ダークマターの位置xが与えられたときの観測データeに対する確率モデルp(e|x)を構築
3. ベイズの定理によりハローの事後分布p(x|e)を求める
4. 事後分布に基づいて、ハローの予測位置に対する期待損失を最小化

例題：Kaggleコンテスト「ダークマターの観測」

- データは360のファイルに分かれており、それぞれが天空を表している
  - 各ファイルには300~720の銀河が含まれている
  - 各銀河の座標はxとy(0~4,200)で与えられる
  - 楕円率はe_1とe_2で与えられている



例題：Kaggleコンテスト「ダークマターの観測」

- 事前分布
  - 各ファイルにはハローが1~3つ含まれる
  - ハローの位置の事前分布として一様分布を使用

$$x_j \sim Uniform(0,4200)$$
$$y_j \sim Uniform(0,4200)\ (j = 1,2,3)$$

  - Timはほとんどの天体には大きなハローが一つだけあると報告
  - 大きなハローの質量を40から180までの対数一様分布に従うと仮定

$$m_{large} \sim \log Uniform(40,180)$$

  - 小さなハローの質量は$\log(20)$に設定

例題：Kaggleコンテスト「ダークマターの観測」

- 各銀河の楕円率は、ハローの位置、銀河とハローの距離、ハローの質量に依存

$$e_i|(x,y) \sim Normal\Big(\sum_{j=\text{ハロー番号}} d_{i,j} m_j f\big(r_{i,j}\big), \sigma^2\Big)$$

  - $d_{i,j}$は接線方向(i番目の銀河の光がj番目のハローで曲げられる方向)
  - m_jはj番目のハローの質量
  - $f\big(r_{i,j}\big)$はハロー jと銀河iのユークリッド距離$r_{i,j}$についての減少関数
    - 大きいハローで

$$f\big(r_{i,j}\big) = \frac{1}{\min(r_{i,j}, 240)}$$

    - 小さいハローで

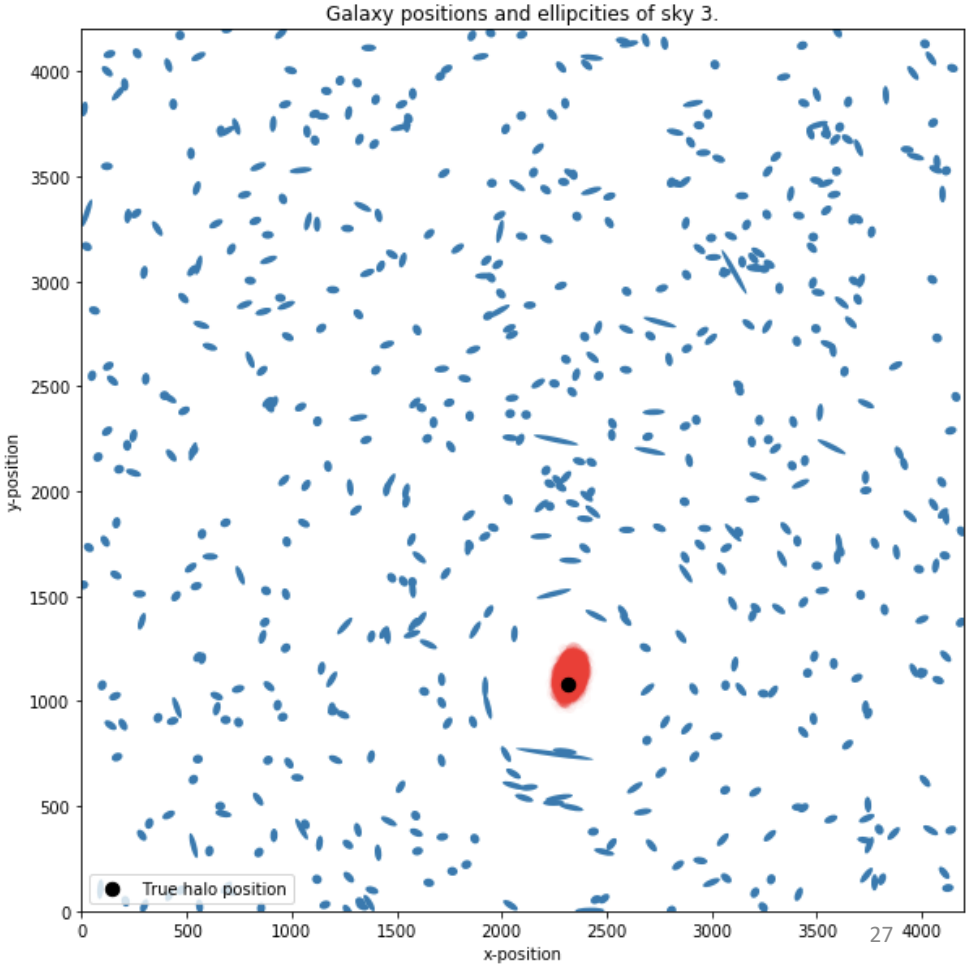
$$f\big(r_{i,j}\big) = \frac{1}{\min(r_{i,j}, 70)}$$

例題：Kaggleコンテスト「ダークマターの観測」

- PyMCで実装してみる(Kaggle_contest.ipynb)

- 事後分布のヒートマップ
- 大きな赤い塊のある場所がハローの存在する事後確率の高い場所
- 黒い点がハローの真の位置

➢ハローの位置を正確に推定



例題：Kaggleコンテスト「ダークマターの観測」

- 損失関数を使って位置を最適化
  - 単純な方法は、平均を使う

```
mean_posterior = t.mean(axis=0).reshape(1, 2)
print(mean_posterior)
```

```
[[ 2324.2292868   1123.41563759]]
```

```
Using the mean:
Your average distance in pixels you are away from the true halo is 42.3177214405
Your average angular vector is 1.0
Your score for the training data is 1.04231772144


Using a random location: [[2712  364]]
Your average distance in pixels you are away from the true halo is 820.025908676
Your average angular vector is 1.0
Your score for the training data is 1.82002590868
```

- 真の位置からそれほど離れていないためスコアが低い
(悪くない予測)

例題：Kaggleコンテスト「ダークマターの観測」

- せっかくなので小さいハローの位置も推定(かなり時間かかります)