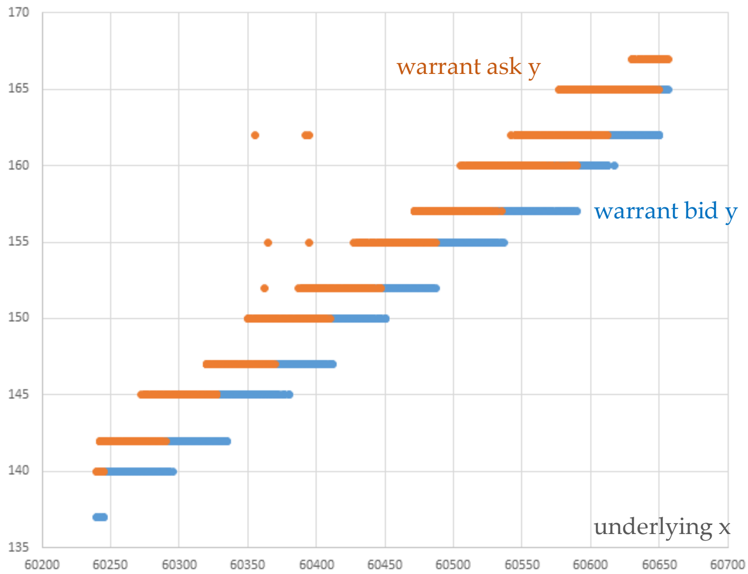
Yubo Securities

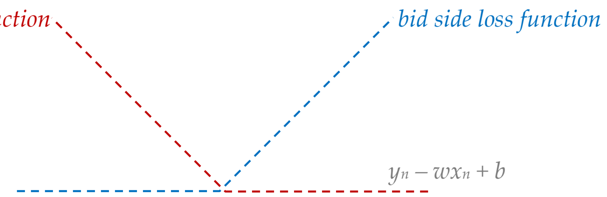
2020 May11

Given 2 hour market data for an underlying transaction price together with the best bid and best ask of its warrant, find a straight line so that the following objective function is minimised.



$$
\begin{aligned}
[w_{opt}, b_{opt}] &= \arg\min_{w,b} L(w,b) \\
&= \arg\min_{w,b} \sum_n s_n f(x_n, y_n \mid w,b) \\
s_n &= \textit{size of} \ \ (x_n, y_n) \\
z_n &= \begin{cases} 1 & \textit{askside} \\ 0 & \textit{bidside} \end{cases}
\end{aligned}
$$

$$
\textit{where} \qquad f(x_n, y_n \mid w,b) = \begin{cases} (wx_n + b) - y_n & z_n \in \textit{askside} \ \ \textit{and} \ \ y_n < wx_n + b \\ 0 & z_n \in \textit{askside} \ \ \textit{and} \ \ y_n \geq wx_n + b \\ y_n - (wx_n + b) & z_n \in \textit{bidside} \ \ \textit{and} \ \ y_n > wx_n + b \\ 0 & z_n \in \textit{bidside} \ \ \textit{and} \ \ y_n \leq wx_n + b \end{cases} \quad \textit{is absolute error (piecewise linear)}
$$



This is ReLU in convolutionary neural network terminology.

Remark

- like logistic regression, this problem is in fact a classification problem rather than a regression problem
- unlike logistic regression, this problem minimises sum of absolute error rather than maximises likelihood
- unlike quadratic optimization, we cannot take differentiation and transform it into linear algebra solution
- sum of absolute error has zero second order derivative everywhere, hence algo depending on L'' cannot apply

My solution is a combination of RanSac and gradient descent, in fact gradient descent for absolute error is simple :

$$
\begin{aligned}
\partial_w L(w,b) &= \sum_n (s_n x_n \cdot 1_{y_n < wx_n + b})^{z_n} \times (-s_n x_n \cdot 1_{y_n > wx_n + b})^{1-z_n} \\
\partial_b L(w,b) &= \sum_n (s_n \cdot 1_{y_n < wx_n + b})^{z_n} \times (-s_n \cdot 1_{y_n > wx_n + b})^{1-z_n}
\end{aligned}
$$

Here is my implementation for mean absolute error and its gradient.

```
auto mae_and_grad(double w, double b)
{
      double mae = 0;
      double dw = 0;
      double db = 0;
      for(const auto& data : dataset)
      {
            double y = w * data.x + b;
            for(const auto& temp : data.y.ask_side) // ask-side
            {
                  if (temp.price < y)
                  {
                        mae += (y - temp.price) * temp.size;
                        dw += temp.size * temp.price;
                        db += temp.size;
                  }
            }
            for(const auto& temp : data.y.bid_side) // bid-side
            {
                  if (temp.price > y)
                  {
                        mae += (temp.price - y) * temp.size;
                        dw -= temp.size * temp.price;
                        db -= temp.size;
                  }
            }
      }
      return std::make_tuple(mae, dw, db);
}
```

In my test, there are 100 trials in coarse search using Ransac, among the coarse search result set, 10-20 best are picked for fine search using gradient descent, the number of iterations is fixed at 10 for gradient descent, the 7 best paths are shown below, indicating the convergence in fitting error. Error is plotted against gradient descent iteration (together with initial guess, there are 11 points for each path).