Statistical Techniques in Robotics (16-831, F14) Lecture#14 (Thursday October 16)

Portfolio Optimization

Lecturer: Drew Bagnell *Scribe: Sezal Jain*[1]

1 Portfolio Optimization - No Regret Portfolio

We want to invest in n different stocks given a set of investment weights w_i s.t. $w_i \in R$, $w_i \geq 0$, and $\sum w_i = 1$. We also know the market returns ratios $r_i = \frac{value^i_{t+1}}{value^i_t}$. So the daily increase in wealth is $w_t^T r_t$, and the total wealth over time is $\prod_t w_t^T r_t$ We want to maximize $\log \Pi w_t^T r_t = \sum \log w_t^T r_t$. We can use use the following algorithm:

$$w_0 \leftarrow \frac{1}{n} \tag{1}$$

$$w_{t+1} \leftarrow Proj[w_t + \frac{\alpha r_t}{w_t^T r_t}] \tag{2}$$

We shall compare the policy to a constantly rebalancing portfolio that maintains a set constant investment ratios. We explain the details in the following.

Here, we control the w_t, the objective function is r_t^i (the return of the i^{th} investment during timestep t) and this is no regret with respect to a constant weight $w^\star$ which is the CRP in return (This means it is a constantly rebalanced portfolio, after every timestep, the assets are moved around such that their starting weight each day is the same - it is not the portfolio where you pick the best asset each day from some oracle).

So for each day $w^T r_t$ is the total return for that day and your total return at the end is $\prod_t w_t^T r_t$.

We will optimize $\sum \log w_t^T r_t$. (As an aside, it seems people actual care about money in a "sort of" logarithmic way, but we are really doing this because it is convenient mathematically).

1) On day 1, buy at some arbitrary set of weights as in equation 1 where w_0 can be set using any a priori knowledge available.

2) We compute interest in w_t

3) Compute gain $\log w_t^T v_t$

4) For each stock i, compute

$$w_{t+1}^i \leftarrow Proj[w_t^i + \frac{\alpha r_t^i}{w_t^T r_t}] \tag{3}$$

Actual regret bound for the optimal rate

$$R \leq \sqrt{F^2 G^2 T} \tag{4}$$

The optimal learning rate is

$$\alpha = \sqrt{\frac{F}{TG}} \tag{5}$$

[1]Notes based on work from previous scribes: Ji Zhang

Filling in the numbers:

1. What is F?

F is largest distance between points in the set. In case of a general simplex

$$F = \sqrt{2}$$

2. What is G?

G is the maximum gradient you can get where gradient is $\frac{r_t}{{w_t}^T r_t}$ If total investment goes to zero i.e ${w_t}^T r_t \to 0$, we cannot get the value of G. But using the assumption $r_{min} \le r_t \le r_{max}$ (i.e no junk bond assumption)

$$|G|^2 \le \sqrt{(\frac{r_{max}}{r_{min}})^2 N} \tag{6}$$

$$\to R \le \sqrt{2(\frac{r_{max}}{r_{min}})^2 NT} \tag{7}$$

$$\alpha = \sqrt{\frac{2}{TN\frac{r_{max}}{r_{min}}}} \tag{8}$$

How can this go wrong?

1. We made a no junk bond assumption

$$r_{min} \le r_t \le r_{max} \tag{9}$$

This assumption fails if there is a catastrophic market crash, but holds for general cases. Intuitively, we can consider the stock price in an economic system won't go more than twice or less than half of its original price in a single day.

2. How do you do the projection?

If $w^i \le 0$, then $w^i = 0$. Now everything doesn't sum to 1. To do the projection correctly, sort the weights, smallest to largest. Then look at $\sum_i max(w_i - a, 0)) = 1$, and find an a such that $\sum w_i = 1$.

This process is called projecting on a simplex, simplex being the weights that sum to one in this case. Renormalizing weights is the wrong way to take a projection as geometrically the method explained above gives the closest point in the simplex to the initial weights.

Gain function: $log(w^T r)$ (take the negative to get the loss)

L_2 Projection (minimize squared difference between point before and after projection)

$$R \leftarrow \sqrt{G} * \sqrt{T} * \sqrt{D(w_0, w*)} \tag{10}$$

3. Transaction costs are not accounted for. The assumption is generally true for big investigation such that transaction cost is neglectable.

4. Moving the market - if you put a lot of money in, you affect the price. We had regret $\sum_t l_t(w_t) - l_t(w*)$. If these losses are a function of decisions you made, then it doesn't mean a

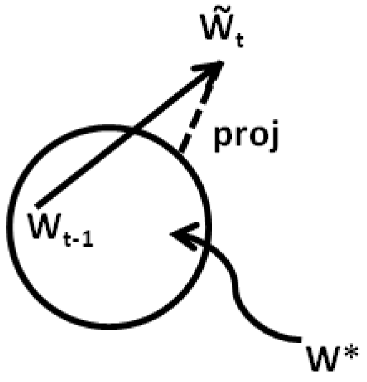

Figure 1: Projection onto a convex set

lot if you have small regret (you just made the best expert perform worse). For example, if one wants to buy fifty percent of the stock of a company, the stock of that company will correspondingly go higher, and as a result, the person won't be able to buy with the same amount of money. This shows up in other robotics applications too: the robot may see something different because it makes actions based on what it sees. For example, the robot decides that a green thing far away is bad and wants to stay as far away from it as possible. Then the robot never learns that maybe green isn't so bad.

All these problems exist in every application of no regret. You're still minimizing your regret, but maybe you aren't minimizing what you hoped you were minimizing. So, no regret is cool, but not magic (ok, maybe a little magic).

Why is projection OK? When you project onto a convex set, you get closer to every point in the convex set.

$$R_t \leq \bigtriangledown l_t^T(w_t - w*) \leq \frac{\alpha G}{2} + [D(\tilde{w}_t, w*) - D(w_{t-1}, w*)] \tag{11}$$

The term $D(\tilde{w}_t, w*) - D(w_{t+1}, w*)$ is always positive (See Figure 1).

Not only does projection not hurt, it could help. We have to get $D(\tilde{w}_t, w*)$ to go down - for this we need the L_2 distance, not L_1. Even if the set is huge, no regret with respect to $D(w_0, w*)$.

$D()$ is the L_2 distance. *Mirror descent* is for other D's besides $(w_0 - w*)^2$.

$\alpha = \sqrt{\frac{F}{GT}}$, if you don't know the horizon, use little t instead of T, α_t instead of a constant α.

$R \leq \sqrt{FGT}$

This is a strongly convex function (bounded by a quadratic) - the quadratic has some steepness.

Every strongly convex function can be written as a convex function.

$f(x) = g(x) + \frac{H}{2} * ||x||^2$, (with f(x) strong)

$abx(x) + ||x||^2$

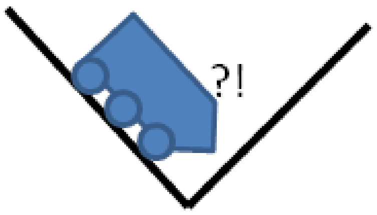

Figure 2: Change in slope is bad

The market problem we just did does not have this property.

$\alpha = \frac{1}{Ht}$, and for this kind of function, $R \leq \frac{G}{2H}(1 + log(T))$, where $\sum \frac{1}{T}$ is bounded by $log(T)$ - notice no dependance on F, so regret grows only logarithmically with the number of time steps.

2 Applications

2.1 Elevation Map from Laser Data

[Projected slides showing terrain as sensed from a plane]

The laser in this example has issues - sometimes it does not penetrate vegetation, and sometimes it appears to go through the ground.

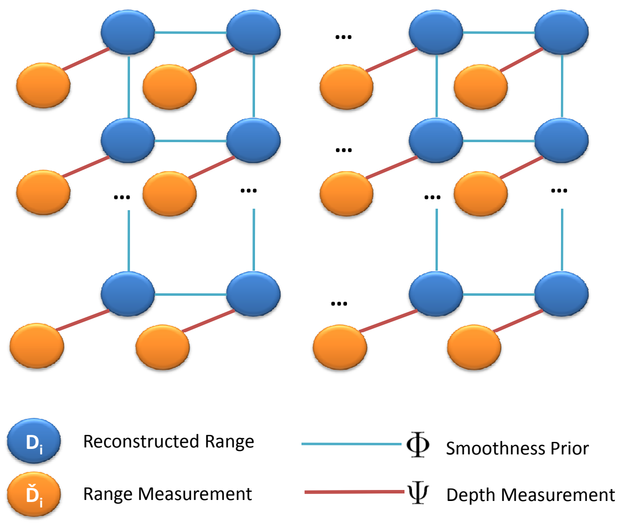

Figure 3: Markov Random Field for Range Sensing

We know that a change in slope is bad for a robot on the ground (See Figure 2). We can represent the ground as an MRF (See Figure 3). The blue nodes represent the height of the ground h_i, and the orange nodes represent the range readings, z_i. The potential functions are defined as:

$$\Phi_{ij} = e^{-\lambda(h_i - h_j)^2} \tag{12}$$

$$\Psi_i = e^{-\sigma(z_i - h_i)^2} \tag{13}$$

where λ is a constant weight placed on the depth measurements.

$$Pr(h) = \frac{1}{Z} \prod_i \Psi_i \prod_{i,j \in neighboringpairs} \Psi_{ij} \tag{14}$$

With Z as the normalizing factor.

Today: $argmax_h Pr(h|z)$

Previously: h_{sample} ˜ $Pr(h|z), E[h_i]$

These two need not agree, but they will here because for a Gaussian distribution, mean = mode.

There is also a third thing (which we won't ever do): $argmax_{h_i} Pr(h_i|z)$, (this is the max marginal distribution)

Now take the log of today's formulation:

$log(Pr(h)) = C - \lambda \sum (h_i - h_j)^2 - \sigma_i \sum (z_i - h_i)^2$

This is convex.

$\bigtriangledown_{l_t}^i = -2\lambda \sum_{j \in neighbori} (h_i - h_j) - 2\sigma(z_i - h_i)$, where the superscript i indicates that the gradient is with respect to h_i.

After applying this, the terrain map [on projected slide] looks smoother.

An alternative way of writing this is:

$$\min_{\bar{x}}(-\log P(\bar{x}|observations)) \to \min_{\bar{x}} \sum_i [\gamma(x_i - h_{i,min})^2 + \lambda \sum_{j \in Neighbors} (|x_i - x_j|)]$$

$$\to (\sum_i 2\gamma(x_i - h_{i,min}) + \lambda \sum_j (sign(x_i - x_j))(\gamma - \alpha)$$

where

$$sign(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

2.2 Generating Super Resolved Range Images

This is based on the paper An Application of Markov Random Fields to Range Sensing by James Diebel and Sebastian Thrun.

In this problem, we have a color image and a depth image, for example taken from an Xtion 3D camera sensor. The depth image as a much lower resolution than the color image. Now the task is to enhance the resolution of the depth image to the same level as the color image. The way of doing this is by assuming that two consecutive pixels in the depth image share the same depth if in the color image they share the same color.

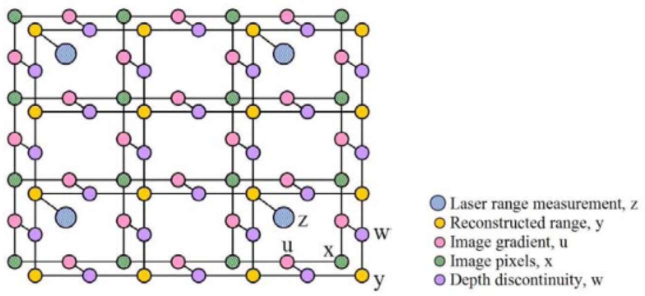

Figure 4: Markov Random Field for the depth and color image

We can use essentially the same MRF model as in Figure 3, but now the z_i nodes only connect to every third (or tenth, etc.) h_i node, but there is a camera pixel for each y_i node with color c_i.

$\delta(color(r_1), color(r_2)) = \sum(p_i - p_j)^2$ for pixels p_i and p_j.

The natural depth measurement potential function is an unnormalized gaussian.

$$\Psi = e^{\frac{-\sigma}{2}(z_i - y_i)^2} \tag{15}$$

From the equation we had before, the depth smoothness prior:

$$\Phi_{ij} = e^{-\lambda_{ij}(y_i - y_j)^2} \tag{16}$$

Taking squared or absolute value has a different effect on the outcome. We now set $\lambda_{ij} = (constant) * e^{-D(c_i, c_j)}$. Here, D reflects how important color is for depth constancy.

Optimize:

$$\max_x(logP(y|z, c)) \tag{17}$$

The gibbs field looks like a giant product of Ψ and Φ functions.

Squared vs Absolute

What difference does it make if we choose

$$\Phi_{ij} = e^{-\lambda_{ij}|y_i - y_j|} \tag{18}$$

While computing the total optimization cost (after taking log):

$$\sum_{i \in N} \frac{\sigma}{2}(z_i - y_i)^2 + \sum_{i,j \in N} \lambda_{ij}|y_i - y_j| \tag{19}$$

Calculating gradients:

$$-\sigma(z_i - y_i) + \lambda_{ij} sign(y_i - y_j) \tag{20}$$

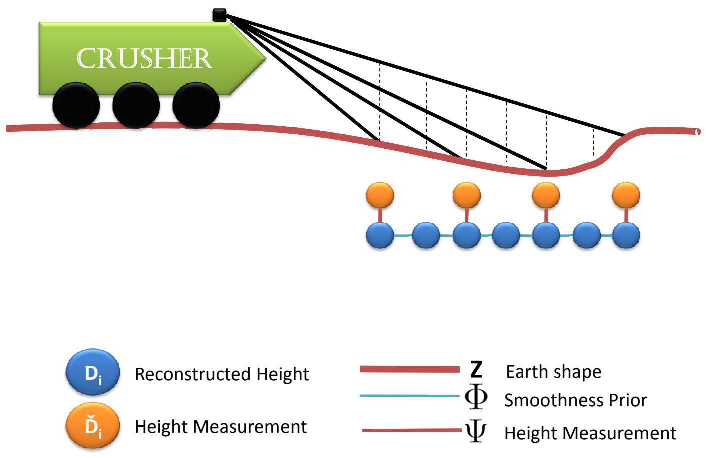

Figure 5: Markov Random Field for Terrain Mapping

2.3 Terrain Mapping for a Ground Robot

The objective in this problem is to estimate the height of the terrain observed with range sensors. Let's assume the set of heights can be described by a function $Z = f(x, y)$ where $Z = 0$ corresponds to the ground level. Figure 5 illustrates a MRF to solve this problem. Note that in this case, Z is defined in 1D.

Similarly to the previous problem, we can define the potential functions as:

$$\Psi = e^{-(\widetilde{D}_i - D_i)^2} \tag{21}$$

$$\Phi = e^{-w_{ij}(D_i - D_j)^2} \tag{22}$$

Moreover, we have the set of constraints $\forall D_i \leq \text{Ray height}_i$ that require a projection function to be satisfied. Note that the set of solutions is convex because it is defined with inequalities.

This assumes no reflections (a puddle could make the robot think there was a hole).

In this case, we get new heights at every time step, so we have a convex set that changes at every time step. This is still OK, because the sequence of convex sets keeps getting smaller.

2.4 Self-supervised Learning

In this example, a robot vehicle is trying to find a specific target using self-supervised Learning.

2.5 Self-training in Nutshell

Here, a robot vehicle navigates in an unknow environment. The robot learns which part of the terrain is drivable and which part is obstacles. The robot looks at the terrain far away and make a guess, then, later on when the robot drives closer, it gets the ground truth. That way the system is trained. The system uses portfolio optimization to adjust weights assigned to different parts of the terrain, and hence converge to a save place to drive on.