COS 511: Foundations of Machine Learning

Rob Schapire
Scribe: John H. White, IV

Lecture #20
4/17/2003

1 Review

Given N stocks you want to learn how to make the most amount of money. Let $\vec{p}_t$ represent how much all the stocks go up and down fractionally on day t and $p_t(i)$ represent how much the i^{th} stock goes up or down fractionally on day t. Let $\vec{w}_t$ represent how much wealth is in all the stocks on day t and $w_t(i)$ represent how much wealth is in the i^{th} stock on day t. As shown in the previous lecture the wealth after T days is $\prod_{t=1}^{T} (\vec{w}_t \cdot \vec{p}_t)$. For Bayes algorithm, this is at least $\frac{1}{N}$ (wealth of best stock). This just boils down to the buy and hold strategy.

2 Constant Rebalanced Portfolio (CRP)

Since the above algorithm is equivalent to the buy and hold strategy let's examine another approach to learning how to make the most money from the stock market. In this example at the end of each day take the wealth and redistribute it into all the stocks based on a given distribution $\vec{b}$. This is called a constant rebalanced portfolio (CRP). Why might this be a good idea? Consider the following example.

Stock 1 - Price stays constant

Stock 2 - Price doubles / halves alternatingly every two days

	Stock 1's p_t	Stock 2's p_t
Day 1	1	$\frac{1}{2}$
Day 2	1	2
Day 3	1	$\frac{1}{2}$
Day 4	1	2
$\vdots$	$\vdots$	$\vdots$

If we let $\vec{b} = \left[\frac{1}{2}, \frac{1}{2}\right]$ then the wealth of the example is

$$S_1 = 1 \tag{1}$$

$$S_2 = S_1 \left(\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{2}\right) = \frac{3}{4} S_1 \tag{2}$$

$$S_3 = S_2 \left(\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2\right) = \frac{3}{2} S_2 \tag{3}$$

$$\vdots$$

$$S_{t+2} = \frac{3}{2} S_{t+1} = \frac{3}{2} \cdot \frac{3}{4} S_t = \frac{9}{8} S_t \tag{4}$$

As Eq. 4 shows every two days the money grows by an $\frac{1}{8}^{th}$. This implies that the money is growing exponentially fast. This example used a "Uniform CRP" because $\vec{b}$ was a uniform probability distribution. As this example shows using CRP's is good.

3 Cover's "Universal Portfolio" (CUP)

Now let's consider when you have many different $\vec{b}$'s and divide the wealth evenly among each one each day. This is Cover's Universal Portfolio algorithm (CUP). Basically CUP is combining many CRP's because each $\vec{b}$ represents a CRP. So on each day t we know the following

$$\left(\mathrm{d}\mu(\vec{b})\right) = \text{amount invested with this } \vec{b} \tag{5}$$

$$\left(\text{wealth using } \vec{b}\right) = \prod_{s=1}^{t-1}\left(\vec{b}\cdot\vec{p}_s\right)\,\mathrm{d}\mu(\vec{b}) \tag{6}$$

$$\left(\text{wealth in stock } i \text{ using } \vec{b}\right) = b\left(i\right)\prod_{s=1}^{t-1}\left(\vec{b}\cdot\vec{p}_s\right)\,\mathrm{d}\mu(\vec{b}) \tag{7}$$

$$(\text{total wealth in stock } i) = \int\left(b\left(i\right)\prod_{s=1}^{t-1}\left(\vec{b}\cdot\vec{p}_s\right)\,\mathrm{d}\mu(\vec{b})\right) \tag{8}$$

$$(\text{fraction of total wealth in stock } i) = \vec{w}_t\left(i\right) = \frac{\int\left(b\left(i\right)\prod_{s=1}^{t-1}\left(\vec{b}\cdot\vec{p}_s\right)\,\mathrm{d}\mu(\vec{b})\right)}{\int\left(\prod_{s=1}^{t-1}\left(\vec{b}\cdot\vec{p}_s\right)\,\mathrm{d}\mu(\vec{b})\right)} \tag{9}$$

From empirical data and analysis when stocks perform as below the algorithm does well.

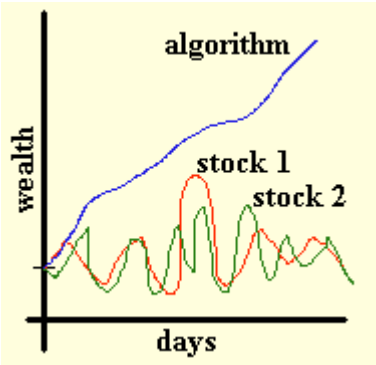

Fig. 1

From theory it is known about this algorithm that

$$(\text{wealth of algorithm after } T \text{ days}) \geq \frac{1}{(T+1)^{N-1}}\,(\text{wealth of best CRP}) \tag{10}$$

Proof: Let $\vec{b}^*$ be the best CRP in hind sight and $\vec{b} = (1-\alpha)\,\vec{b}^* + \alpha\vec{z}$ where $\alpha \in [0,1]$ and $\vec{z}$ is a CRP in the set $\triangle$ of all CRP's. By doing this $\vec{b}$ defines a volume of CRP's within $\vec{b}^*$. Since $\vec{b}$ invests $1-\alpha$ of its money the same as $\vec{b}^*$, it must make at least $1-\alpha$ as much as $\vec{b}^*$ on each day. Thus,

$$\left(\text{wealth of } \vec{b}\right) \geq (1-\alpha)^T \left(\text{wealth of } \vec{b}^*\right) \tag{11}$$

So by Eq. 11 we know that the neighborhood of CRP's defined by $\vec{b}$ has

$$\text{wealth} \geq (1-\alpha)^T \left(\text{wealth of } \vec{b}^*\right)$$

Now the volume of the CRP's defined by $\vec{b}$ is

$$\begin{aligned} & \texttt{Vol}\left(\left\{(1-\alpha)\,\vec{b}^* + \alpha\vec{z} \;:\; \vec{z} \in \triangle\right\}\right) && (12)\\ = \;& \texttt{Vol}\left(\{\alpha\vec{z} : \vec{z} \in \triangle\}\right) && (13)\\ = \;& \alpha^{N-1}\texttt{Vol}\left(\triangle\right) && (14)\end{aligned}$$

Therefore at least α^{N-1} of the CRP's have

$$\text{wealth} \geq (1-\alpha)^T \left(\text{wealth of } \vec{b}^*\right)$$

So for the algorithm,

$$(\text{wealth of algorithm}) \geq \alpha^{N-1}\,(1-\alpha)^T \left(\text{wealth of } \vec{b}^*\right) \tag{15}$$

Now by letting $\alpha = \frac{1}{T+1}$,

$$\begin{aligned} (\text{wealth of algorithm}) \;&\geq\; \frac{1}{(T+1)^{N-1}}\left(1-\frac{1}{T+1}\right)^T \left(\text{wealth of } \vec{b}^*\right) && (16)\\ &\geq\; \frac{1}{(T+1)^{N-1}}\left(\frac{1}{e}\right)\left(\text{wealth of } \vec{b}^*\right) && (17)\end{aligned}$$

Therefore Eq. 10 is proved true within the constant $\frac{1}{e}$. Eq. 10 can be proved exactly by choosing a "better" neighborhood. Consult the literature on the topic to see the proof. ■

The theory for this algorithm is for the worst case. In practice algorithms exist that do better empirically but their theory says they should do worse. In practice "Uniform CRP's" do as well as the others.

4 Thoughts on Experimentation

4.1 Test Carefully

Programs written for machine learning research and experimentation can be very difficult to debug. Use small "made up" data sets or small "real" data sets so that the results can be calculated by hand to verify the programs. Compare the output of the program to known theoretical results to see if the program is behaving properly. Also since the programs are run on computers be aware of numerical issues arrising from limited space to represent numbers.

4.2 Number of Data Sets

When using data sets in analysis try to use 20 to 30 different data sets. This gives the program credibility in showing that it is not an aberration that only works on one or two data sets.

4.3 Limited Amount of Data

When using data sets that have a small number of examples the technique of *10 Fold Cross Validation* should be used. This involves dividing the data up into 10 equally sized sets of examples. Then for each set of 10% of the examples train on the remaining 90% of examples and then test on the 10%. This involves running the algorithm 10 times but by doing this and taking the average of the error rates on each run better results are obtained.

4.4 Avoid Overfitting Test Set

Don't try all the algorithms you can think of to try and get a better result on the error of the test set. By just getting the best error on the test set you can overfit it and cause the generalization error to be increased. This can also be caused by using algorithms with many parameters and then tweaking the parameters to give the best test error. Instead try to guess reasonable values for the parameters and stick with those values. Also the data can be divided up into training, validation, and test sets. Use the training and validation sets to analyze the algorithm but then only use the test set to report the performance of the algorithm.

4.5 Automate

Try and automate as much as possible. This gets rid of errors introduced by humans when trying to remember what files they have and have not copied, what test they performed, etc. Also by automating as much as possible a record of what occurred is preserved. This way the experiments are repeatable by not only you but others reviewing your work. Also when using randomness use seeding values and remember them. This way the randomness is repeatable when the experiments are reviewed.

4.6 Beware of Real Computers

Machine learning experiments are often very time consuming and require many trials. Don't forget that computers are real and they do crash. Try to design your program to write out intermediate results and recover from crashes. This way when a crash occurs all the program's work will not be lost.

4.7 Experiments Have a Point

Experiments should be planned ahead of time and have the purpose of enforcing your work. Make sure they relate to what you are trying to say. Also use experiments to isolate what makes your work better or newer than other work.

4.8 Graphs

Use graphs and pictures whenever possible to display results. Lists of numbers upon numbers are not pleasing to read.

4.9 Established Software

Use software packages that are already written to aid in your work or even as benchmarks against which to compare your work. Two such packages to consider are WEKA[1] and MLC++[2].

5 Exact Identification

Up until now we assumed that the learning algorithm got data examples passively. What if the learner was able to actively choose examples? Consider trying to learn an exact function c that classifies the examples. The algorithm has a black box that represents the function c. The algorithm is also allowed to ask questions to the black box and it uses these results to create a functionally equivalent function to function c. There are two queries that can be asked. The first query is a Membership Query. This query asks "What is the value of function c on example x?" This query can be thought of as a set and it is asking is x a member of the set.

To be continued... (in the next lecture)

[1] http://www.cs.waikato.ac.nz/ml/weka/
[2] http://www.sgi.com/tech/mlc/