Advanced Topics on Cloud Computing **Spring 2021**

Lecture 1: Resource Planning

Lecturer: Dr. Georgios Paschos *Amazon*

This lecture focuses on *Resource Planning*: reserving an amount of resources for future use. Pertinent questions in the context of cloud computing include: (1) How many servers to install in a datacenter in order to meet a future demand? (2) What is the storage, processing power and networking capability that would suffice for a customer needs throughout a year? (3) How should we adjust the resources for a cloud service in order to meet changing demands? In this lecture we begin with the newsvendor problem, the archetypical resource planning question. We then discuss how forecasting affects planning, and present an *elastic* planning technique based on online learning.

1.1 The newsvendor problem

We purchase y units of a perishable good (think of newspapers, or croissants) at unit cost c, and sell at unit price p; both c, p are known. There will be a demand for X units, where X is unknown at purchase phase, and modeled by a random variable with known distribution. Given unsold goods perish at the end of the day, *what is the choice of y that maximizes the expected total reward?*

The newsvendor problem is one of the most fundamental problems in operation research and applied economics. It sets the basic dilemma of any resource planning problem: invest too much and waste resources, or invest too little and loose sales.

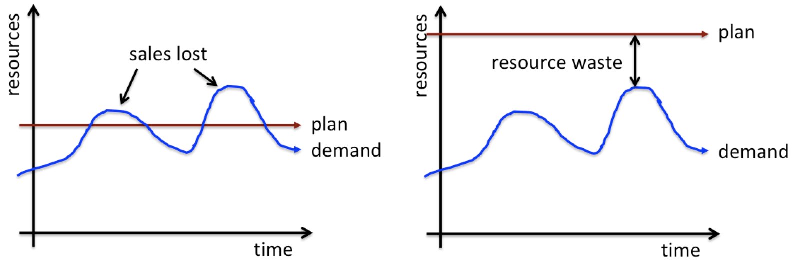

Figure 1.1: Resource planning.

We will focus on the simple case where the resource comes in continuous quantities and the demand distribution is differentiable everywhere.[1]

Newsvendor problem:

$$\max_{y\geq 0} p\mathbb{E}[\min(y,X)] - cy$$

The first term of the objective captures the expected profit from sales, while the second term captures the cost of the entire investment. Suppose F is the cumulative distribution of the demand, i.e., $F(x) = \mathbb{P}(X \leq x)$, and $\varphi(x)$ its derivative, which we assume it exists everywhere.

Lemma 1.1 *The optimal newsvendor investment is given by the critical fractile formula:*

$$y^* = F^{-1}\left(\frac{p-c}{p}\right).$$

[1]When the resource comes in units, we need to further consider the optimization on a grid, however, in low dimensions it is possible to use the presented continuous formulation and check which rounding to nearest integer works best. When the demand has probability jumps, a more detailed proof is needed.

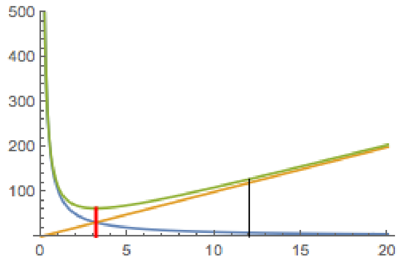

Figure 1.2: Selection of optimal investment in the newsvendor model: (blue) opportunity cost by expected missed sales, (orange) investment cost, (green) total cost, (red) min-cost investment, which corresponds to max-reward.

Proof: By the Law of iterated expectations:

$$\begin{aligned}\mathbb{E}[\min(y,X)] &= \mathbb{P}(X \le y)\mathbb{E}[\min(y,X)|X \le y] + \mathbb{P}(X > y)\mathbb{E}[\min(y,X)|X > y] \\ &= \int_{x \le y} x\varphi(x)dx + (1-F(y))y\end{aligned}$$

Its derivative simplifies to:

$$\frac{\partial \mathbb{E}[\min(y,X)]}{\partial y} = y\varphi(y) + (1-F(y)) - y\varphi(y) = 1 - F(y). \tag{1.1}$$

Note that the newsvendor objective is $U(y) = p\mathbb{E}[\min(y,X)] - cy$. Then using (1.1) we have $U'(y) = p - c - pF(y)$, and $U''(y) = -p\varphi(y) \le 0$. Hence, U is concave, and a necessary condition for y^* to be optimal is $U'(y^*) = 0$. This yields $F(y^*) = \frac{p-c}{p}$, which inverted gives the result. ■

The newsvendor model is used to determine the resource plan that minimizes cost in expectation, when the demand distribution is known and the costs are linear. In this case, the optimal investment is given by $F^{-1}\left(\frac{p-c}{p}\right)$.

1.2 Planning to a forecast

The newsvendor model assumed the demand is characterized by a known distribution. In reality, future demand is predicted by means of forecasting. Below we introduce demand forecasting and provide some basic tools. A beginner's introduction can be found in [2].

1.2.1 Demand forecasting

We are given $t-1$ historical values of demand, $x_1, x_2, \ldots, x_{t-1}$, and we are asked to forecast the future demand x_t. We denote our forecast with F_t. We may evaluate our forecast by making N rounds of forecasts $F_t, \ldots, F_{t+N-1}$, and compare those to the actual demand $x_t, \ldots, x_{t+N-1}$. At each round, we compute the *error* E_t (or residual) using the equation: $E_t = F_t - x_t$. A helpful metric of forecast success is the ***Mean Absolute Percentage Error*** (*MAPE*), defined as:

$$MAPE = 100 \cdot \frac{\sum_{i=t}^{t+N-1} |E_i/x_i|}{N}.$$

The problem of *demand forecasting* is to develop a method to systematically produce F_t that minimizes *MAPE*.

Apart from evaluating MAPE, it is worthy to look at other ***residual diagnostics***. The simplest test is the average $\overline{E} = 1/N \sum_{i=t}^{t+N-1} E_i$. A forecast is said to be unbiased if $\overline{E}$ tends to 0 as N increases. A biased forecast tends to err in a given direction and can be improved by de-biasing techniques. The histogram of residuals also provides valuable information for optimization, and often fits a Normal or a t-student distribution. Finally, the

Auto-Correlation Function (ACF) of the residual sequence provides information to the forecaster as to whether there exist seasonal components in the signal or other correlations. If there are correlations between residuals, then there is information left in the residuals which should be used in computing forecasts. For example, this information can be used to decompose the signal into seasonal and non-seasonal parts.
Naive forecast. The simplest possible forecast is to use the last value in the sequence:

$$F_t = x_{t-1}.$$

This forecaster does not learn from the structure of our time-series, but it may perform well in situations of rapid and unexpected growth. It can be used as a baseline.
Mean estimator. This forecast assumes that past and future actuals are drawn independently from the same stochastic model.

$$F_t = \frac{1}{t-1}\sum_{i=1}^{t-1} x_i.$$

In absence of periodic (or other type) structure, the mean estimator can turn out to be a good forecast, and is often used due to its simplicity. If the actuals are seasonal, the mean estimator can also be adjusted to seasonality, e.g., we may have one estimator for weekdays and another for weekends.
Exponential smoothing. Sometimes we desire a prediction that is between the two extremes of naive and mean estimator. Let $\alpha \in (0,1)$ be a constant, we define the exponential smoothing forecaster to be:

$$F_t = \alpha x_{t-1} + (1-\alpha)\alpha x_{t-2} + \cdots + (1-\alpha)^{t-2}\alpha x_1.$$

Notice that this forecaster becomes the naive forecast as $\alpha \to 1$ and the mean estimator as $\alpha \to 0$. Values of α close to 1 allow the forecaster to "forget" quickly.
Auto-Regressive Moving Average (ARMA). One way to apply linear regression to forecasting is by training ϕ on a p-order auto-regressive model:

$$F_t = \mathbb{E}[F_t] + E_t + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p},$$

where E_t is assumed to be zero-mean Gaussian. Further, it is possible to merge this model with a moving average model, which similarly to the mean estimator averages past residuals:

$$F_t = \mathbb{E}[F_t] + E_t + \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{j=1}^{q} \theta_j E_{t-j},$$

The best ARMA model can be trained on the data set by minimizing a forecasting error metric over ϕ, θ.

ARMA models are very popular statistical methods for time series forecasting, and can be further enhanced using differencing/integration (ARIMA), and seasonality (SARIMA). An important question for an ARMA model is the correct choice of p and q. This can be done either by checking stationarity of residuals or by using automated algorithms such as auto-ARIMA.

Other interesting techniques left out of scope are neural networks with Long-Short Memory (LSTM), nearest neighbors, multi-variate forecasting, and variational forecasting such as DeepAR [4].

> The best forecasting model varies according to the problem, but flexible models such as ARIMA and DeepAR are always a good place to start.

1.2.2 Robust planning

Sometimes, the objective of planning is simply to ensure that the resources are "enough". Mathematically, such an objective can be formulated as a *chance constraint*, targeting to limit the probability of resource outage to a small controllable value. Suppose a distributional forecast representing the future resources needed by K cloud computing users. The future demand of user k is expressed in the following way:

$$x_k = \mathbb{E}[F_k] + E_k,$$

where F_k is the forecast demand of user k and the residual $E_k \sim \mathcal{N}(0, \sigma^2)$, i.i.d. across users. Let ϵ be a small and controllable constant.
Robust planning: Find the minimum resources y that keep the outage probability at most ϵ:

$$P(y_k < x_k) \leq \epsilon \Leftrightarrow P(E_k > y_k - \mathbb{E}[F_k]) \leq \epsilon \Leftrightarrow Q\left(\frac{y_k - \mathbb{E}[F_k]}{\sigma}\right) \leq \epsilon \Leftrightarrow y_k \geq \mathbb{E}[F_k] + \sigma Q^{-1}(\epsilon)$$

where, $Q(.)$ is the tail distribution of standard normal. Intuitively, $\sigma Q^{-1}(\epsilon)$ represents a buffer for accomodating uncertainty. This methodology generalizes to any stochastic optimization problem, and allows us to model chance constraints as linear constraints [5].

Resource pooling is the idea of serving clients from a common pool of resources. Cloud computing technology allows to build large datacenters, and then serve users flexibly by sharing resources among them. To demonstrate the benefit, consider as before the case we want to cover the demand of each of K clients with same probability, but this time we allow resource pooling. It then suffices to satisfy:

$$\begin{aligned} P\left(\sum_k y_k < \sum_k x_k\right) \leq \epsilon &\Leftrightarrow P\left(\sum_k E_k > \sum_k y_k - \sum_k \mathbb{E}[F_k]\right) \leq \epsilon \\ &\Leftrightarrow Q\left(\frac{\sum_k y_k - \sum_k \mathbb{E}[F_k]}{\sqrt{K}\sigma}\right) \leq \epsilon \\ &\Leftrightarrow \sum_k y_k \geq \sum_k \mathbb{E}[F_k] + \sqrt{K}\sigma Q^{-1}(\epsilon) \end{aligned}$$

where we used the fact that $\sum_k E_k \sim \mathcal{N}(0, K\sigma^2)$. We observe that the total required buffer reduced from $K\sigma Q^{-1}(\epsilon)$ to $\sqrt{K}\sigma Q^{-1}(\epsilon)$. For $K = 100$, this represents a 90% reduction in the required buffer.

The uncertainty risk is best hedged by aggregating multiple plans.

1.3 Elastic Planning with Online Convex Optimization

We present an elastic planning methodology that can be used to adjust to a changing demand. The methodology is designed to yield the best performance even when the future demand is engineered by an adversary to hurt our plans.

At time $t = 1, \ldots, T$ a policy π chooses an investment plan $y_t^\pi \in \mathcal{Y}$, buying $y_{t,k}^\pi$ units of good k. The investment returns a reward from an unknown function $f_t(y_t^\pi)$. Set $\mathcal{Y}$ is convex and functions $f_t, t = 1, 2, \ldots, T$ are convex with bounded gradients $\|\nabla f_t\| \leq B$, and we assume that function f_t is revealed to the policy at the end of round t.

Cloud computing example

Suppose $y_{t,k}^\pi \in [0, X_{\max}]$ denotes the amount of computing power planned for user k, and $x_{t,k} \in [0, X_{\max}]$ the unknown demand of that user. The reward function adds a quadratic penalty whenever the demand exceeds the planned resource:

$$f_t(y_t^\pi) = -c\sum_{k=1}^{K} y_{t,k}^\pi - \sum_{k=1}^{K}(x_{t,k} - y_{t,k}^\pi)^2 \mathbb{1}_{\{x_{t,k} > y_{t,k}^\pi\}},$$

where c is a weighting constant. For sufficiently small c, the above function is maximized by setting the plan just above the demand of each user. Unfortunately, the demand is unknown, hence the algorithm has to additionally learn how the demand evolves.

We define the *static regret* of π as the cumulative difference between the policy reward and the reward of the *optimal investment in hindsight*:

$$\texttt{Reg}^\pi(T) = \sum_{t=1}^{T} f_t(y^*) - \sum_{t=1}^{T} f_t(y_t^\pi)$$

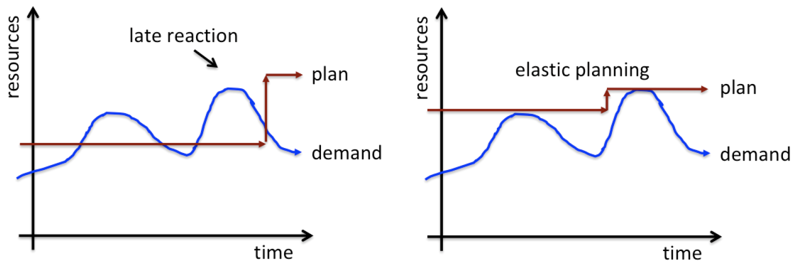

Figure 1.3: Elastic resource planning for cloud computing.

where $y^* = \arg\max_{x\in\mathcal{Y}} \sum_{t=1}^{T} f_t(x)$ is the investment in $\mathcal{Y}$ that maximizes reward when knowing all functions (a.k.a. in hindsight). The regret will be a metric of how well our algorithm is performing, and we will look for a policy that minimizes regret, or at least yields a regret which is sublinear to the horizon. If the latter is true for an algorithm, it follows that as the horizon T increases, the average reward of the algorithm is as good as the best action in hindsight.

The unknown functions can be interpreted as stochastic, or adversarial. In the latter case, an adversary observes policy choice y_t^π, and then selects a function f_t that worsens the regret. Although in most planning problems there are no real adversaries, taking such an approach produces all-rounder algorithms that are robust to non-stationary fluctuations, and also work well in situations of stationarity.

Problem solution

The standard policy for solving OCO problems is the projected Online Gradient Ascent (OGA) algorithm–proposed in [6]. It chooses an initial vector $y_1 \in \mathcal{Y}$ and iterates using:

$$\begin{aligned} z_{t+1} &= y_t + \eta_t g_t, \\ y_{t+1} &= \mathcal{P}_{\mathcal{Y}}(z_{t+1}), \text{ for } t = 1, \dots, T-1 \end{aligned} \tag{1.2}$$

where (i) g_t is the gradient of f_t at y_t, (ii) η_t is the stepsize, and (iii) $\mathcal{P}_{\mathcal{Y}}(y)$, is the Euclidean projection of y on set $\mathcal{Y}$. Let R be the diameter of $\mathcal{Y}$ and B be an upper bound on gradients, such that $\|g_t\| \leq B, t = 1, \dots, T$.

Theorem 1.2 (Regret of OGA) *Fix stepsize $\eta = R/(B\sqrt{T})$, the regret of OGA satisfies:*

$$\mathtt{Reg}^{OGA}(T) \leq RB\sqrt{T}.$$

Proof: Adapted from [7]. For any point $y^* \in \mathcal{Y}$ we can expand the norm to write

$$\|z_{t+1} - y^*\|^2 = \|y_t + \eta g_t - y^*\|^2 = \|y_t - y^*\|^2 + 2\eta {g_t}^T (y_t - y^*) + \eta^2 \|g_t\|^2$$

and by the non-expansiveness of the Euclidean projection, we can derive a bound on the distance of an algorithm iterate from the best action in hindsight:

$$\begin{aligned} \|y_{t+1} - y^*\|^2 &= \|\mathcal{P}_{\mathcal{Y}}(z_{t+1}) - y^*\|^2 \leq \|x_{t+1} - y^*\|^2 \\ &= \|y_t - y^*\|^2 + 2\eta {g_t}^T (y_t - y^*) + \eta^2 \|g_t\|^2. \end{aligned}$$

Summing telescopically over the horizon, we obtain:

$$\|y_T - y^*\|^2 \leq \|y_1 - y^*\|^2 + 2\eta \sum_{t=1}^{T} ({g_t}^T (y_t - y^*)) + \eta^2 \sum_{t=1}^{T} \|g_t\|^2.$$

Since the LHS is positive, re-arranging terms we obtain:

$$\sum_{t=1}^{T} ({g_t}^T (y^* - y_t)) \leq \frac{R^2}{2\eta} + \frac{\eta T B^2}{2}. \tag{1.3}$$

Since f_t are concave, for any $y \in \mathcal{Y}$ it holds $f_t(y) - f_t(y_t) \leq {g_t}^T(y - y_t)$ and combining we deduce:

$$\texttt{Reg}^{OGA}(T) = \sum_{t=1}^{T}(f_t(y^*) - f_t(y_t)) \leq \sum_{t=1}^{T}({g_t}^T(y^* - y_t)) \overset{(1.3)}{\leq} \frac{R^2}{2\eta} + \frac{\eta T B^2}{2}.$$

Plugging in $\eta = R/B\sqrt{T}$, we obtain the result. ■

In [8] it was shown that a simple adversary using linear functions can force any policy π to suffer a regret:

$$\texttt{Reg}^{\pi}(T) \geq \frac{RB}{2\sqrt{2}}\sqrt{T},$$

which shows that OGA has optimal regret less a small constant. The best achievable regret can be improved if the adversary is restricted to specific choices, e.g., using only strongly convex functions [9], while it may deteriorate if the OCO problem includes adversarial time-average constraints, see [10, 11, 12]. A generalization of OGA is the online mirror descent which can improve the constants of the regret related to the dimensions of set $\mathcal{Y}$ [7].

The online gradient is a simple online planning algorithm that is optimal in highly volatile environments.

OGA for Elastic Planning

The last step is to derive the gradient algorithm for our example. We begin by computing the directional derivative:

$$\frac{\partial f_t}{\partial y_k} = -c + 2(x_{t,k} - y_k)\mathbb{1}_{\{x_{t,k} > y_k\}} = \begin{cases} 2\delta_k(y_k) - c & \text{if } y_k < x_{t,k}, \\ -c & \text{otherwise} \end{cases}$$

where $\delta_k(y_k) = x_{t,k} - y_k$ is the amount of resource more than y_k that would suffice to meet the demand in the previous slot. The gradient is simply $\nabla f_t(y) = \left(\frac{\partial f_t}{\partial y_1}, \ldots, \frac{\partial f_t}{\partial y_K}\right)^T$, and hence the algorithm is updating plan for user k in the following way:

$$\begin{cases} y_{t+1,k} = (y_{t,k} + \eta(2\delta_k(y_k) - c))^+ & \text{if } y_k < x_{t,k}, \\ y_{t+1,k} = (y_{t,k} - \eta c)^+ & \text{otherwise.} \end{cases}$$

1.4 Quiz

You are given a dataset that contains two months of Amazon tweets between 26/02/2015 and 22/04/2015. We are working on a webserver application that processes these tweets, and which requires cloud resources. We would like to plan such resources for the next week time at an hourly granularity. The cost of cloud resources is \$0.1 per tweet, and there is a user experience penalty equal to $p(\delta) = \$0.5\delta^2$, where $\delta \geq 0$ is the number of resources we must urgently subscribe in case we are short.

Specifically, we are interested in the last week of data from 13/04/2015 until 19/04/2015, which we will use as a testing week for our purposes. The task is to design a model that proposes an amount of resources to reserve hourly for this week. You are asked to compare two techniques:

(a) An online gradient technique.

(b) An ARIMA forecast paired with a stochastic optimization approach.

Report back your findings, and ideas for further improvements if any.

References

[1] S. Axsäter, *Inventory Control*, Springer International Publishing, 3rd edition, 2015.

[2] R. J. Hyndman and G. Athanasopoulos, *Forecasting Principles and Practice*, OTexts: Melbourne, Australia. OTexts.com/fpp2, 2nd edition, 2018.

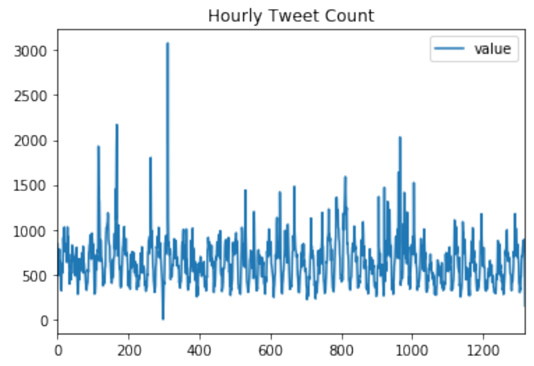

Figure 1.4: Public dataset with tweets from 26/02/2015 to 22/04/2015.

[3] T. Gneiting, F. Balabdaoui, and A. Raftery, *Probabilistic forecasts, calibration and sharpness*, Springer International Publishing, hal-00363242, 2007.

[4] `https://ts.gluon.ai/`

[5] D. Bertsimas, D. Brown, and C. Caramanis, *Theory and applications of robust optimization*, SIAM review 53.3, 464–501, 2011.

[6] M. Zinkevich, *Online Convex Programming and Generalized Infinitesimal Gradient Ascent*, 2003.

[7] V. Belmega, P. Mertikopoulos, R. Negrel, and L. Sanguinetti, *Online Convex Optimization and No-Regret Learning: Algorithms, Guarantees, and Applications*, arXiv:1804.04529v1, Apr. 2018.

[8] J. Abernethy, P. Bartlett, A. Rakhlin, and A. Tewari, *Optimal strategies and minimax lower bounds for online convex games*, 2008.

[9] E. Hazan and K. Satyen, *Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization*, The Journal of Machine Learning Research 15.1, 2489–2512, 2014.

[10] S. Mannor, J. Tsitsiklis, and J. Yuan Yu, *Online Learning with Sample Path Constraints*, Journal of Machine Learning Research, 10.3, 2009.

[11] M. Neely and H. Yu, *Online convex optimization with time-varying constraints*, arXiv preprint arXiv:1702.04783, 2017.

[12] N. Liakopoulos, A. Destounis, G. Paschos, T. Spyropoulos, and P. Mertikopoulos, *Cautious regret minimization: Online optimization with long-term budget constraints*, ICML, pp. 3944-3952. 2019.