Aggregating Web Offers to Determine Product Prices

Rakesh Agrawal
Microsoft Research, Search Labs
rakesh.agrawal@microsoft.com

Samuel Ieong
Microsoft Research, Search Labs
samuel.ieong@microsoft.com

ABSTRACT

Historical prices are important information that can help consumers decide whether the time is right to buy a product. They provide both a context to the users, and facilitate the use of prediction algorithms for forecasting future prices. To produce a representative price history, one needs to consider all offers for the product. However, matching offers to a product is a challenging problem, and mismatches could lead to glaring errors in price history. We propose a principled approach to filter out erroneous matches based on a probabilistic model of prices. We give an efficient algorithm for performing inference that takes advantage of the structure of the problem. We evaluate our results empirically using merchant offers collected from a search engine, and measure the proximity of the price history generated by our approach to the true price history. Our method outperforms alternatives based on robust statistics both in tracking the true price levels and the true price trends.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Algorithms, Experimentation

Keywords

Data Aggregation, Time Series Analysis

1. INTRODUCTION

The Internet has become one of the most important sources of information for consumers researching for their next purchase. While there are many e-commerce websites dedicated to helping consumers decide what product to buy and where to buy it, few sites exist to help with the question of *when* to buy the product. In [1, 2], the authors proposed a system that addresses this need by providing users with product price history and making recommendations of buy or wait depending on price forecasts and consumer preferences. In their experiments, product prices are obtained through a data vendor. In this paper, we investigate the question of how to aggregate web offers to determine representative product prices, thus circumventing the explicit dependence on data vendors.

If the set of all offers correctly matched to the product were given as input, determining the product prices would have been straight-forward: one can simply take the average of the prices (or the minimum, depending on the application). Unfortunately, matching offers to products is known to be a difficult problem, and despite much work in the area (reviewed further in Section 2), it is inevitable that a matching algorithm will make mistakes. We are thus interested in the following question:

> Given a set of offers matched to a product, how do we determine a representative price history for the product, *allowing for some of the matches to be possibly incorrect.*

Note that to produce a good and representative price history, it is not necessary (although it would be sufficient) for an algorithm to identify all of the incorrect matches. For example, if we are interested in the average product price, an incorrectly matched offer with a price close to the average will be mostly harmless. Errors are introduced when we fail to identify incorrect matches with prices significantly different than the average (or in the case of minimum, significantly lower than the true minimum).

As the main challenge in aggregating web offers is due to incorrect offers with prices that are outliers, could the problem be solved using robust statistics [12]? Focusing on determining the average price, for example, could we replace the simple average of product prices by the trimmed mean, where the top and bottom $x\%$ of the data points are discarded before the average is computed, or by the median, often considered to be robust to outliers? It turns out that these solutions are inadequate for two reasons.

First, errors made by matching algorithms are typically not random. A common form of error is confusing one product with another. For example, confusing two models of TVs made by the same manufacturer, or confusing the accessories of a product with the product itself. This form of errors leads to entire sets of offers being (mis)matched to the same product. Another common form of error is due to

KDD'12, August 12–16, 2012, Beijing, China.

products that are configurable. For example, a digital SLR camera may be packaged together with different lenses. The different configurations may be sold at very different prices. Under both form of errors, trimmed mean will remove some offers from both the most and the least expensive sets, but the resulting price will likely remain a mixture of multiple set of offers, and the final price is not representative of a single product. The median will select an offer from one particular set of offers, but as the composition of offers changes over time, it may select an offer from a different set at in the future, leading to inconsistencies and creating discontinuity in the price history when none is present.

Second, a direct application of robust statistics fails to take advantage of the sequential nature of the data. Using all the matched offers across time can help in two ways. First, determining whether an offer is correctly matched or not at an isolated time point is difficult; a valid price can easily be confused with an outlier. By taking into account the entire price history of an offer, we improve our ability to decide whether the offer itself is an outlier. Also, in trying to disambiguate between multiple clusters of prices (with similar number of offers), we can use the historical prices in deciding which cluster is most likely to be correct.

To address these shortcomings, we propose to model offer prices by a generative process motivated by linear Gaussian processes [21]. We model the true product prices as unobserved latent variables that follow some linear dynamics. We associate each offer with a Bernoulli variable that determines whether the offer is correct for this product. If it is correct, the offer prices will be drawn according to a Gaussian distribution with a mean equal to the underlying product prices (and variance to be learned, as different offers exhibit different price volatilities); otherwise the offer prices will be sampled from a background distribution. This model allows us to take advantage of the sequential nature of the data, and by explicitly modeling whether an offer is correctly matched, the solution is sensitive to the type of errors typically made by matching algorithms described above. To ensure efficiency of our learning algorithm, we design an inference algorithm that takes advantage of an independence assumption and allows us to achieve a quadratic speed-up over standard Kalman filter. The technique may be of independent interest.

The rest of the paper is organized as follows. We review related work in offer matching, Kalman filters, and deciding when to buy in Section 2. We then present a probabilistic model of offer prices in Section 3. We consider how to learn the model parameters in Section 4. To speed-up the learning algorithm, we introduce a technique that makes use of the independence assumptions in the model in Section 5. We evaluate the performance of algorithm using merchant offers collected from a search engine over six months in Section 6. We conclude with our main findings and directions for future research in Section 7.

2. RELATED WORK

Matching offers to products is a central problem in ecommerce [16]. Offers for products are often partly structured (including URLs, prices, and sometimes Universal Product Codes (UPCs)) and partly unstructured (offer titles and textual descriptions). There has been much work in the database and data mining communities on matching structured records to other structured records, including record linkage [8, 19, 20, 25], entity resolution [3, 23], and duplicate detection [7, 22]. There has also been some work on matching unstructured text to structured records that uses techniques based on natural language processing [18], segmentation [16, 17], and clustering [4]. Our work builds on these past works and assumes that an initial matching of offers to products has been performed, and takes the output and creates representative product prices for the products. As part of the process, we need to identify incorrect matches where the prices are significantly different than others. While prices can be used as part of the matching process, to our knowledge, there has been no work that uses the sequentiality of the prices in determining the correctness of a match. Our work complements the existing work and can be used to handle matching scenarios where some attribute values of a record are slowly changing over time.

The main modeling technique in this paper is by modeling the offer prices as observations generated from a linear dynamical system (also known as linear Gaussian processes). A good summary of this research area from the machine learning perspective is given in [21]. The study of linear dynamical dates back to early work in signal processing, with the Kalman filter being one of the seminal work in the area [14]. Learning the parameters to a linear dynamical system, also known as system identification, has also been well studied and a review is given in [9]. Many extensions to Kalman filters have been proposed, for example, the extended Kalman Filter [26] and particle filters [11, 13, 15]. While these work relax various assumptions on the linearity of the process and provide efficient algorithms for learning, their models do not consider the possibility of erroneous observations.

In our work, we augment the classical linear dynamical model by allowing for observations to be possibly incorrect. This is related to the robust Kalman filter [27] and Kalman filter with intermittent observations [24]. For the latter work, using a cutoff criteria, one can treat a single outlier as a missing observation. However, in both lines of works, the basic modeling units are individual data points, hence the learned model will not take into account the entire offer (consisting of all offer prices) in deciding whether an observation is an outlier. Closer in spirit to our work is the switching state-space model proposed in [10]. One can view the background prices in our model as a separate state-space model, and that the switching variable of [10] controls whether a set of observations is generated by the background prices or the product prices. However, the learned model will instead group the correct and incorrect observations by time; we are interested in grouping them by offer. As a final note, the technique proposed in Section 5 to speed up the computation of the Kalman Filter under independent observation assumptions is novel to our knowledge, and could be of independent interest.

There has been recent work that investigates the question of helping users to decide when to buy a product [1, 2]. The present study extends this line of work and considers how to obtain representative product prices automatically by aggregating web offers. The history can then be presented to users in response to product queries, be fed as input to forecasting algorithms.

3. MODEL

We now present a generative model of offer prices motivated by linear Gaussian processes. Informally, the model

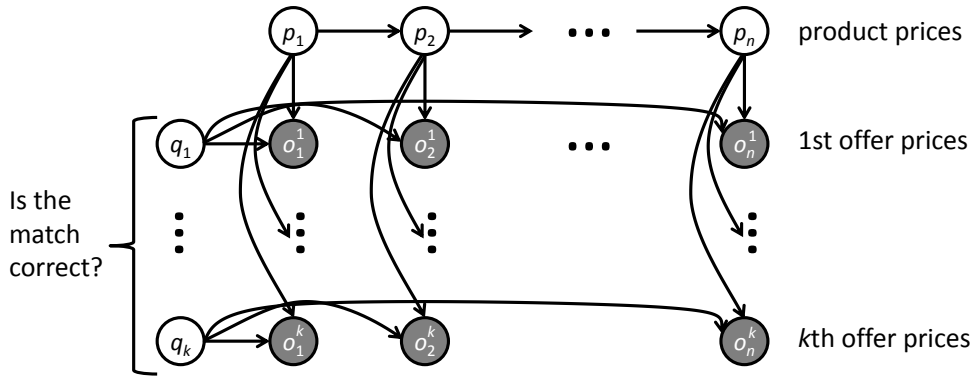

Figure 1: Graphical representation of the generative model of offer prices. The shaded nodes are the observations, which include all of the offer prices $\mathbf{o}^1, \mathbf{o}^2, \ldots, \mathbf{o}^k$. The unshaded nodes are latent variables in the model, which include the product prices p and offer status q. The parameters of the model, not shown here to reduce clutter, are learned to maximize likelihood of the observations.

postulates that there is a set of true but unobserved prices for the product, one for each time step, and that the prices evolve according to some yet-to-be-learned linear dynamics. A set of offers are purportedly matched to the product. An offer is either correctly matched, in which case its prices are drawn according to a distribution parameterized by the product prices, or incorrectly matched, in which case its prices are drawn according to some background price distribution. Our goal is to learn the the underlying prices and the correctness of the matching treating the offer prices as observations.

Formally, we consider a discrete-time generative model as illustrated in Figure 1. Let the true but *unobserved* product prices be denoted $\mathbf{p} = \{p_1, p_2, \ldots, p_n\}$. The prices are generated by a linear Gaussian process, determined by four parameters: μ_0, σ_0^2, α, and σ_α^2, where

$$p_1 \sim N(\mu_0, \sigma_0^2)$$
$$p_t \sim N(\alpha p_{t-1}, \sigma_\alpha^2) \qquad \text{for } 1 < t \leq n$$

and $N(\mu, \sigma^2)$ denotes a Gaussian distribution with mean μ and variance σ^2.

Let the number of offers matched to the product be k, which may vary from product to product. For the i-th offer, its prices $\mathbf{o}^i = \{o_1^i, o_2^i, \ldots, o_n^i\}$ are generated as follows. First, an *unobserved* Bernoulli variable q_i, which we call the *offer status*, is drawn according to parameter π_i. If q_i equals 1, the offer is correctly matched, and the *observed* offer price o_t^i at time t is drawn according to a Gaussian distribution with mean p_t and variance σ_i^2, which we call the *observation variance* of offer i. On the other hand, if q_i equals 0, the offer is incorrectly matched, and the *observed* offer price o_t^i at time t is drawn according to a Gaussian distribution with mean μ_b and variance σ_b^2 that corresponds to some background distribution of prices. We treat this background distribution as exogenously given, and do not learn them in the model as this distribution is estimated using all offers matched to the products of a given category.

Summing up, for each offer i,

$$q_i \sim \text{Bernoulli}(\pi_i)$$
$$o_t^i \sim \begin{cases} N(p_t, \sigma_i^2) & \text{if } q_i = 1 \\ N(\mu_b, \sigma_b^2) & \text{otherwise} \end{cases} \qquad \text{for } 1 \leq t \leq n \quad .$$

Under this model, the probability of a set of observed prices $\mathbf{O} = \{\mathbf{o}^1, \mathbf{o}^2, \ldots, \mathbf{o}^k\}$ can be expressed as a function of the parameters $\Theta = (\mu_0, \sigma_0^2, \alpha, \sigma_\alpha^2, \{\pi_i\}_{i=1}^k \{\sigma_i^2\}_{i=1}^k)$ and background mean and variance μ_b and σ_b^2 by summing out the latent variables, namely,

$$P(\mathbf{O}; \Theta) = \int \sum_{\mathbf{q} \in \{0,1\}^k} P(\mathbf{O}, \mathbf{p}, \mathbf{q}; \Theta) d\mathbf{p}$$

where

$$P(\mathbf{O}, \mathbf{p}, \mathbf{q}; \Theta) = P(p_1; \mu_0, \sigma_0^2) \prod_{t=2}^{n} P(p_t | p_{t-1}; \alpha, \sigma_\alpha^2)$$
$$\prod_{i=1}^{k} \left(\left(\pi_i \prod_{t=1}^{n} P(o_t^i | p_t; \sigma_i^2) \right)^{q_i} \left((1 - \pi_i) \prod_{t=1}^{n} P(o_t^i; \mu_b, \sigma_b^2) \right)^{(1-q_i)} \right) \quad (1)$$

The parameters of the model can be learned, once for each product, by treating the offer prices as observations; this will be discussed further in Section 4. The learned parameters together with the observations can then be used to jointly infer the most likely product prices. In addition, the parameters π_i's, corresponding to the likelihood of an offer being correctly matched, can be used to determine the set of correct matches by comparing its value to a cutoff threshold. The offers deemed correctly matched by the model can then be used to compute the average or the minimum product prices as desired.

Note that in a real running system, offer prices may be missing at times for various reasons, hence we need to make provisions in the generative model to allow for missing observations. We assume that with some fixed probability r an observation may be missing, independent of whether the offer is correctly matched or not. This assumption allows us to simplify the derivation of the inference procedure, as the probability of a missing observation can be factored out from the likelihood function in Equation (1).

4. INFERENCE AND LEARNING

At runtime, given the offer prices, we can learn the values of the parameters of the model by maximizing the likelihood function. As the model consists of both latent and observed variables, we can generalize the Expectation-Maximization (EM) framework [6] to learn the parameters. Following standard EM derivation (see, e.g., Chapter 9 of [5]),

- In the E-Step, we infer the distribution of the latent variables given the parameters, i.e., we compute

$$Q(\mathbf{p}, \mathbf{q}) = P(\mathbf{p}, \mathbf{q}|\mathbf{O}; \Theta)$$

- In the M-Step, we find the values of the parameters that maximizes the likelihood, treating the probability estimates Q as given, i.e., we compute

$$\Theta^{new} = \arg\max_{\Theta} \int \sum_{\mathbf{q}\in\{0,1\}^k} Q(\mathbf{p}, \mathbf{q}) \log P(\mathbf{O}, \mathbf{p}, \mathbf{q}; \Theta) d\mathbf{p}$$

Due to the interactions between the latent variables $\mathbf{q}$ and $\mathbf{p}$, an exact inference is intractable in the E-step. Instead, we consider an approximate inference procedure by variational methods, discussed further in Section 4.1. The optimization of parameters in the M-step are fairly straightforward, and for completeness we present them in Section 4.2. The overall approach is motivated by the learning algorithm for switching state-space models [10], with adaptations to account for the differences in the model.

4.1 The Variational E Step: Inference

Given the observations $\mathbf{O}$, the distribution of the latent product price variables $\mathbf{p}$ and latent offer status variables $\mathbf{q}$ are *not* independent; this can be verified from the graphical structure of the model (Figure 1) that these variables are not d-separated given the observations. In order to perform efficient inference, we apply variational approximation in the E-step.

Specifically, we focus on the following structural approximation to $Q^{(j)}(\mathbf{p}, \mathbf{q})$ that decouples the variables $\mathbf{p}$ and $\mathbf{q}$, *viz.*

$$P(\mathbf{p}, \mathbf{q}|\mathbf{O}; \Theta) = Q(\mathbf{p}, \mathbf{q}) = Q(\mathbf{p})Q(\mathbf{q}) \ .$$

We then find the factors $Q(\mathbf{p})$, $Q(\mathbf{q})$ that minimizes the KL-divergence to the correct non-independent distribution $P(\mathbf{p}, \mathbf{q}|\mathbf{O}; \Theta^{old})$. Using the general result from Chapter 10 of [5], this can be found by iteratively solving the following system until convergence:

$$\log Q(\mathbf{q}) = \mathbb{E}_{\mathbf{p}}[\log P(\mathbf{O}, \mathbf{p}, \mathbf{q}; \Theta)] + \text{constant} \quad (2)$$

$$\log Q(\mathbf{p}) = \mathbb{E}_{\mathbf{q}}[\log P(\mathbf{O}, \mathbf{p}, \mathbf{q}; \Theta)] + \text{constant} \quad (3)$$

where the constants ensure that the factors $Q(\mathbf{p})$ and $Q(\mathbf{q})$ integrates to one.

From Equations (2) and (1), we can verify that $\log Q(\mathbf{q})$ can be expressed as an exact factorization over the individual offer statuses, i.e.,

$$\log Q(\mathbf{q}) = \sum_{i=1}^{k} \log Q(q_i)$$

where

$$\log Q(q_i) = \mathbb{E}_{\mathbf{p}}[\log P(q_i|\mathbf{O}, \mathbf{p}; \Theta)] + \text{constant} \ .$$

Solving for $Q(q_i)$ for each offer i, we find that

$$\begin{aligned} \log Q(q_i = 1) &\propto n \log \frac{1}{\sqrt{2\pi}\sigma_i^2} - \sum_{t=1}^{n} \mathbb{E}_{\mathbf{p}} \left[\frac{(o_t^i - p_t)^2}{2\sigma_i^2} \right] \\ \log Q(q_i = 0) &\propto n \log \frac{1}{\sqrt{2\pi}\sigma_b^2} - \sum_{t=1}^{n} \frac{(o_t^i - \mu_b)^2}{2\sigma_b^2} \end{aligned} \quad (4)$$

which has the natural interpretation that the log odds between $Q(q_i = 1)$ and $Q(q_i = 0)$ is precisely the ratio of the log likelihoods of offer i being generated by the product prices or through the background distribution.

From Equations (3) and (1), we can verify that $\log Q(\mathbf{p})$ can be expressed as an exact factorization according to the linear dynamics that underlie the product prices, i.e.,

$$\log Q(\mathbf{p}) = \log Q(p_1) + \sum_{t=2}^{n} \log Q(p_{t-1}, p_t)$$

where

$$\log Q(p_1) = \mathbb{E}_{\mathbf{q}}[\log P(p_i|\mathbf{O}, \mathbf{q}; \Theta)] + \text{constant}$$

and for $1 < t \leq n$,

$$\log Q(p_{t-1}, p_t) = \mathbb{E}_{\mathbf{q}}[\log P(p_t|p_{t-1}, \mathbf{O}, \mathbf{q}; \Theta)] + \text{constant}$$

As that the factorization preserves the structure of the dynamics, we can apply the forward-backward algorithm (also known as Kalman filter and smoother) for inferring the latent variables given the parameters. To take into account the offer statuses $\mathbf{q}$ (over which we are taking expectations), it is necessary and sufficient to divide the observation variance σ_i^2 of offer i by $Q(q_i = 1)$ in the algorithm (the same has been observed in [10]). This has the effect of decreasing the influence of offer i to the product prices if the likelihood of offer i being correctly matched is low.

Note that just as in the standard derivation of inference for linear dynamical systems, instead of determining the distribution $Q(\mathbf{p})$ exactly, we only determine the required expectations under the distribution. We follow the notations and derivations from [5] with suitable modifications. Formally, let $\mathbf{1}_k$ denote a column vector of ones of length k, and $\mathbf{1}_{k\times k}$ denote a square matrix of all ones of size $k \times k$ (*not* to be confused with the identity matrix of size $k \times k$). Let Σ be a $k \times k$ matrix that corresponds to the modified observation variances, with the i-th diagonal entry equal to $\frac{\sigma_i^2}{Q(q_i=1)}$, and zero elsewhere. Let $\mathbf{o}_t$ denotes all observed offer prices at time t. The forward stage, corresponding to the Kalman filter, can be computed iteratively from $t = 1$ to n, where for $t = 1$,

$$\begin{aligned} K_1 &= \sigma_0^2 \mathbf{1}_k^T (\sigma_0^2 \mathbf{1}_{k\times k} + \Sigma)^{-1} \\ \mu_1 &= \mu_0 + K_1(\mathbf{o}_1 - \mathbf{1}_k \mu_0) \\ v_1 &= (1 - K_1 \mathbf{1}_k)\sigma_0^2 \\ c_1 &= N(\mathbf{o}_1 | \mathbf{1}^k \mu_0, \sigma_0^2 \mathbf{1}_{k\times k} + \Sigma) \end{aligned}$$

and for $1 < t \leq n$,

$$\begin{aligned} p_{t-1} &= \alpha^2 v_{t-1} + \sigma_\alpha^2 \\ K_t &= p_{t-1}\mathbf{1}_k^T(p_{t-1}\mathbf{1}_{k\times k} + \Sigma)^{-1} \\ \mu_t &= \alpha\mu_{t-1} + K_t(\mathbf{o}_t - \mathbf{1}_k\alpha\mu_{t-1}) \\ v_t &= (1 - K_t\mathbf{1}_k)p_{t-1} \\ c_t &= N(\mathbf{o}_t|\mathbf{1}_k\alpha\mu_{t-1}, p_{t-1}\mathbf{1}_{k\times k} + \Sigma) \end{aligned}$$

The backward stage, corresponding to the Kalman smoother, can be computed iteratively from $t = n-1$ down to 1, where

$$\begin{aligned} j_t &= v_t\alpha/p_t \\ \hat{\mu}_t &= \mu_t + j_t(\hat{\mu}_{t+1} - \alpha\mu_t) \\ \hat{v}_t &= v_t + j_t^2(\hat{v}_{t+1} - p_t) \end{aligned}$$

For both the variational E-step and the subsequent M-step, the required expectations are given by

$$\begin{aligned} \mathbb{E}_{\mathbf{p}}[p_t] &= \hat{\mu}_t \\ \mathbb{E}_{\mathbf{p}}[p_t^2] &= \hat{\mu}_t^2 + \hat{v}_t \\ \mathbb{E}_{\mathbf{p}}[p_t p_{t-1}] &= j_{t-1}\hat{v}_t + \hat{\mu}_t\hat{\mu}_{t-1} \end{aligned}$$

To sum up, in the variational E-step, we alternate between computing $Q(\mathbf{p})$ and $Q(\mathbf{q})$ until convergence based on the system of equations described in Equations (2) and (3). The factor $Q(\mathbf{p})$ is not determined explicitly, but rather we use the forward-backward algorithm to determine the required expectations. The factor $Q(\mathbf{q})$ is determined explicitly using the log odds described in Equation (4).

Note that the most expensive computation step in the E-step is inverting the $k \times k$ matrices $(\sigma_0^2\mathbf{1}_{k\times k} + \Sigma)$ and $(p_{t-1}\mathbf{1}_{k\times k}+\Sigma)$ in computing the Kalman gain matrices K_t at the forward stage. Note that these are *not* diagonal matrices and hence a direct computation of the inverses will take time $O(k^3)$. However, one can take advantage of the structure of these matrices to avoid the need to take inverses explicitly. This is discussed further in Section 5.

4.2 The M Step: Parameter Learning

In the M-step, we learn the parameters $\Theta = (\mu_0, \sigma_0^2, \mu_a, \sigma_a^2, \{\pi_i\}_{i=1}^k\{\sigma_i^2\}_{i=1}^k)$ by maximizing the likelihood of the observed data. The optimization can be solved analytically by taking the derivative of each of the parameters and setting it to zero. For completeness, the updates to the parameters are

$$\begin{aligned} \mu_0 &= \mathbb{E}_{\mathbf{p}}[p_1] = \hat{\mu}_1 \\ \sigma_0^2 &= \mathbb{E}_{\mathbf{p}}[p_1^2] - \mathbb{E}_{\mathbf{p}}[p_1]\mathbb{E}_{\mathbf{p}}[p_1] = \hat{v}_1 \\ \alpha &= \frac{\sum_{t=2}^n \mathbb{E}_{\mathbf{p}}[p_t p_{t-1}]}{\sum_{t=2}^n \mathbb{E}_{\mathbf{p}}[p_{t-1}^2]} \\ \sigma_\alpha^2 &= \frac{1}{n-1}\sum_{t=2}^n\Big(\mathbb{E}_{\mathbf{p}}[p_t^2] - 2\alpha\mathbb{E}_{\mathbf{p}}[p_t p_{t-1}] + \alpha^2\mathbb{E}_{\mathbf{p}}[p_{t-1}^2]\Big) \end{aligned}$$

and for $1 \leq i \leq k$,

$$\begin{aligned} \pi_i &= Q(q_i = 1) \\ \sigma_i^2 &= \frac{1}{n}\sum_{t=1}^n\Big((o_t^i)^2 - 2o_t^i\mathbb{E}_{\mathbf{p}}[p_t] + \mathbb{E}_{\mathbf{p}}[p_t^2]\Big) \end{aligned}$$

Putting it altogether, the learning algorithm is given in Algorithm 1.

Algorithm 1: EM Algorithm for learning parameters to generative model of offer prices

input : Offer prices matched to a product $\mathbf{O}$
output: Parameters to the model Θ

Initialize the values of Θ (random or by heuristics);
repeat
  Variational E-step:
  repeat
    Compute required expectations according to $Q(p)$ using forward-backward algorithm;
    Compute $Q(q)$ according to Equation (4);
  until *convergence*;
  M-step: Update Θ according to Section 4.2;
until *convergence*;
return Θ;

5. KALMAN FILTER SPEEDUP

As mentioned in Section 4.1, the dominant computation in the learning algorithm is inverting the matrices needed in the forward stage of the inference algorithm. We now consider how this step can be sped up by avoiding the explicit need to compute the inverse.

As this technique may be of independent interest, we derive it in a case more general than the one considered in Section 4.1. Let s_0 be some scalar quantity, $\mathbf{c} = [c_1, c_2, \ldots, c_k]$ be a column vector of length k, and D be a diagonal matrix of size $k \times k$ with the diagonal elements being $s_1, s_2, \ldots, s_k$. Consider the matrix M formed by

$$M = \mathbf{c}s_0\mathbf{c}^\mathbf{T} + D \ .$$

For convenience of notation, let $c_0 = 1$. We can show by induction the following lemmas related to the matrix M.

Lemma 1. *The determinant of M equals*

$$|M| = \sum_{0\leq m\leq k}\Big(\prod_{0\leq\ell\leq k,\ell\neq m} s_\ell\Big)c_m^2 \ .$$

Lemma 2. *The inverse of M equals*

$$(M^{-1})_{ij} = \frac{1}{|M|}\begin{cases} \sum_{0\leq m\leq k, m\neq i}\Big(\prod_{0\leq\ell\leq k,\ell\neq i,m} s_\ell\Big)c_m^2 & \text{for } i = j \\ \Big(\prod_{0\leq\ell\leq k,\ell\neq i,j} s_\ell\Big)c_i c_j & \text{for } i \neq j \end{cases} .$$

Lemma 3. *The product of vector $a^T = [s_0c_1, s_0c_2, \ldots, s_0c_k]$ and M equals*

$$(a^T M^{-1})_i = \frac{1}{|M|}\Big(\prod_{0\leq\ell\leq k,\ell\neq i} s_\ell\Big)c_i \ .$$

where $(a^T M^{-1})_i$ is the i-th entry of the vector $(a^T M^{-1})$.

By setting $\mathbf{c} = \mathbf{1}_k$, $D = \Sigma$, and $s_0 = \sigma_0^2$ or p_{t-1} as appropriate, together with Lemma 3, we can avoid taking the matrix inverse explicitly, and instead compute the Kalman gain filter directly in $O(k)$ time. Note that the evaluation of the determinant of M using Lemma 1 will only take $O(k)$ time, as one can first pre-compute the product $\left(\prod_{0\leq\ell\leq k} s_\ell\right)$ in $O(k)$ time and obtain the required product in the summand by division, thus avoiding taking $O(k)$ time to evaluate each of the k summands. Likewise this applies to the evaluation in Lemma 3.

6. EVALUATION

We now present an empirical evaluation of our technique using real product and offer data obtained from a commerce search engine.

6.1 Experimental Setup

We obtain a set of products related to televisions from a commerce search engine for which we want to create price histories. The products include both televisions and accessories such as remote controls and mounts. For some of these products, we manage to obtain its Universal Product Code (UPC), a unique identifier.

For the products with UPCs, we obtain a set of offers that are matched to each of them according to the search engine. We have also obtained the offer prices over a five-month period from mid July 2011 to mid December 2011. We filter out any product that include offers with prices that exhibit a regime change (prices going up or down by more than 50% from day to day), as these are often due to two offers being confused as one, and are not proper candidate for evaluation.

For some of these offers, we manage to obtain their UPCs, and we treat an offer with a UPC that is equal to that of the product as ground truth. There are certainly limitation to this ground truth set—it is possible that an offer with a missing UPC could be a correct match; it is also possible that the UPC is incorrect. However, obtaining human judgments for whether an offer is correctly matched to the product is a difficult and costly process, and an attempt to obtain judgment via Mechanical Turk has resulted in very noisy labels, hence we settle on employing UPCs as ground truth. To eliminate some of the mistakes due to incorrect UPCs, we filter out cases where the ground truth includes offers for which their prices differ by over 100%, as these are signs that some of the UPCs are incorrect. We also require that a product has to have there are at least three offers in the ground truth set.

To identify aberrations in the ground truth set, we compare the daily average prices according to the ground truth, and the daily average prices according to all offers with prices between the lowest and the highest prices among the offers in the ground truth. *A priori*, there is no reason based on a price argument that an offer within this range is not a correct match. Therefore, if many of the offers are missing, it is an indication that the set of offers in the ground truth for this product is too small. As the usefulness of our metrics depend on a sampling argument (see next subsection), the results measured on products for which this happens are less reliable. We remove from consideration products where the daily averages according to the two computation differs by more than 5% at the peak. After these processing steps, our evaluation is conducted over a total of 700 products.

6.2 Metrics

For each product, we compute the average product price history and the minimum product price history according to the ground truth set. We measure the difference between these price histories (both average and minimum) and the ones generated algorithmically via two metrics.

Our first metric is the mean scaled absolute difference (*MASE*), designed to measure how closely the generated price history tracks the price level of the true price history. Given the price history $\mathbf{p}^*$ computed from ground truth and the algorithmic price history $\mathbf{p}$, the MASE is defined as:

$$MASE(\mathbf{p}^*, \mathbf{p}) = \frac{1}{n} \sum_{t=1}^{n} \frac{|p_t^* - p_t|}{p_t^*} \ .$$

The lower the MASE, the better the performance of the algorithm. We denote the MASE for the average price history as Avg-MASE and that for the minimum price history as Min-MASE. We report the numbers averaged across all products.

Our second metric is the mean absolute change difference (*MACD*), designed to measure how closely the generated price history tracks the price trends in the true price history. Given the price history $\mathbf{p}^*$ computed from ground truth and the algorithmic price history $\mathbf{p}$, the MACD is defined as:

$$MACD(\mathbf{p}^*, \mathbf{p}) = \frac{1}{n-1} \sum_{t=2}^{n} \left| \frac{p_t^* - p_{t-1}^*}{p_{t-1}^*} - \frac{p_t - p_{t-1}}{p_{t-1}} \right| \ .$$

This metric captures the average difference between the change in prices according to each history in relative terms. The lower the MACD, the better the performance of the algorithm. We denote the MACD for the average price history as Avg-MACD and that for the minimum price history as Min-MACD. We report the numbers averaged across all products.

As a final note, we discuss how the choice of the ground truth set affects the usefulness of these metrics. If the ground truth set contains all and only correctly matched offers, the metrics will measure the desired difference between the true price histories and the algorithmically genearted ones. Assuming that the correct matches in the ground truth set is only sampled (uniformly) from among all of the correct matches, and that the prices of the correctly matched offers are distributed according to a Gaussian distribution, the average price history based on the ground truth is an unbiased estimator of the true average price history, and hence we expect Avg-MASE and Avg-MACD to be good metrics, provided the samples are sufficiently dense. On the other hand, the minimum price history based on the ground truth will systematically overestimate the true minimum price history. Hence, the Min-MASE and Min-MACD should be interpreted with caution. However, given the importance of minimum price histories for consumers in deciding when to buy a product, we decide to keep these metrics in the evaluation.

6.3 Choices for Our Algorithms

There are two approaches to computing the average price history using the probabilistic model proposed in this paper. First, we can use the inferred latent prices $\mathbf{p}$ computed in the final iteration of the variational E-step in Algorithm 1. In the experiment, we label this method as Latent. Alternatively, we can use the parameters $\{\pi_i\}_{i=1}^k$ that correspond to the likelihoods of the offers being correct matches. Together with a cutoff threshold τ, we can select all offers with $\pi_i > \tau$, and take the average over these offers. In the experiment, we label this method as Match.

To compute the minimum price history, the inferred latent prices are not directly applicable. Therefore, we follow the paradigm of Match as described in the preceding paragraph, and instead of taking the average we take the minimum after offers are selected.

In all of our experiments, we have selected τ to be 0.6.

After inspecting the values of the parameters $\{\pi_i\}_{i=1}^k$ for a number of products, we notice that the values are typically very close to 0 or 1. Hence, the choice of τ has minimal effect on the performance of Match.

6.4 Algorithms for Comparison

We compare our approach to three alternatives in our experiments for the evaluation on average price histories. These algorithms are (labels in parentheses):

- Trimmed mean (Trim): for each day, the top and bottom 10% of the offer prices are removed before the average is taken.
- Median (Median): for each day, the median among all offer prices is chosen.
- Postprocessing with Kalman filter and smoother (Post KF): form a price history by taking the daily average of all offer prices, then apply the Kalman filter and smoother to the resulting price history treating it as observations.

We choose the trimmed mean and the median as they are common and popular alternatives to the simple average, and they are considered more robust to outliers. We also choose the Kalman filter and smoother applied as a post-processing step after the average is taken, as it can help to smooth away anomalous averages that are caused by the occasional outliers.

For the evaluation on minimum price histories, the median no longer makes sense. Instead, we select the price of the offer closest to the 10-th percentile as the product price. We call this method as the Trimmed Min (Trim) as it is analogous to removing the top and bottom 10% of the offer prices before taking the minimum. We continue to evaluate the Kalman filter and smoother as a post-processing step applied to the minimum taken over all offer prices.

Finally, as baseline, we consider using the entire set of matched offers to generate the price history. In the experiment, we label that as All.

6.5 Results

The performances of our algorithms, the baseline, and the alternatives are presented in Table 1. Note that while these percentages may appear small, they do matter to prospective consumers. For example, for a $2,000 TV (a typical price for a brand-named LCD TV), a MASE of 5% corresponds to a difference of $100 on average between the true price and the reported price each day. Such errors can significantly decrease credibility of the system. Likewise, a MACD of 1% corresponds to a difference of $20 on average between the true price change and the reported price change. The errors quickly add up and may lead to poor recommendations for consumers who are deciding between buying now or wait.

For each metric, we present its value, followed by % difference from the baseline in the next row. Recall that under all four metrics, the lower the value, the better the performance; improvements over baseline will give a *negative* % difference. Our main findings are:

- Match is the best algorithm overall. The differences in performance under all four metrics between Match and the baseline All are statistically significant under paired t-tests, with $p < 0.1$ for both MASE measures and $p < 0.005$ for both MACD measures. The magnitude of improvements are higher for the minimum price history, and under MACD that measures price trends.

Metric	All	Trim	Median	Post KF	Latent	Match
Avg-MASE	3.48%	3.77%	4.49%	3.58%	4.84%	**3.34%**
	-	*+8%*	*+29%*	*+3%*	*+39%*	*-4%*
Avg-MACD	0.63%	0.63%	1.00%	0.69%	0.67%	**0.58%**
	-	*0%*	*+59%*	*+9%*	*+6%*	*-9%*
Min-MASE	4.59%	5.14%	N/A	4.85%	N/A	**4.27%**
	-	*+12%*	-	*+6%*	-	*-7%*
Min-MACD	0.80%	0.94%	N/A	0.80%	N/A	**0.59%**
	-	*+18%*	-	*0%*	-	*-26%*

Table 1: Overall Results of All Algorithms Under Four Metrics. The best performing approaches are highlighted in bold.

- The baseline All is a difficult baseline to beat. This should perhaps be not too surprising as UPCs constitute an important signal commonly used in matching algorithms. Indeed, none of the alternatives, Trim, Median, or Post KF, has outperformed All on any of the metrics. This shows that the type of errors made by All cannot be addressed by simply applying more robust statistics, or by smoothing away an occasional outliers.
- Although Match and Latent employ the same probabilistic model, Latent performed far worse than Match. This was initially quite puzzling to us. Through an error analysis, we determine the causes were due to the product price dynamics; this is discussed further in Section 6.7.

6.6 Factor Analysis

To better understand when we could expect Match to do well, and under what conditions Match will outperform All and vise versa, we analyze different groupings of the products to identify factors that are highly correlated with performance. We identify two factors—fraction of offers matched in the ground truth, and the difference in prices among matched offers—as the two most influential factors. Their effects on Avg-MASE and Avg-MACD are presented in Figure 2. Their effects on Min-MASE and Min-MACD exhibit the same trends and are omitted due to space limitations. The key observations are:

- The fraction of offers that are matched to a product in the ground truth set is negatively correlated with MASE and MACD for both Match and All, i.e., higher fraction of offers matched lead to better performance. This is unsurprising and can be explained by the experimental setup, as an increasing fraction of offers in the ground truth set increases the likelihood that an offer matched to the product is correct and will contribute to the average product price.

  From the system design viewpoint, an insight from this analysis is that there exists a cross-over point at around 70% when All starts to outperform Match. As most matching algorithm in ecommerce utilizes UPC as one of the features for matching, one can readily

Figure 2: Primary factors that influence the relative performance between Match and All.

use the fraction of offers with matched UPC as a feature, and only apply Match when the fraction falls below some threshold, and derive a hybrid approach that should outperform both.

- The difference between the correctly matched offers with the highest and lowest prices is positively correlated with MASE and MACD for both Match and All, i.e., larger differences lead to worse performance. This points to the increased difficulty of creating representative price histories as the differences in offer prices increase. There is no clear cross-over point like the preceding factor, hence a hybrid approach is unlikely to help with this scenario.

  On examining further the set of products with large differences, we discover that there are often fundamental issues with the matching associated with such products. This is discussed further in the next subsection.

6.7 Error Analysis

Examining the cases for which our method performs poorly, we note that a common cause is due to the existence of multiple sets of offers, each with distinct price ranges, being mapped to the same product, and in the ground truth set, offers from two or more sets are deemed correct. This problem often also leads to large differences in the prices of the matched offers, a problem noted before. This leads to poor performance for Match, as the probabilistic model often learns parameters that corresponds to one set of offers, and Match produces an average price history that corresponds only to the set selected.

We find it intuitively unlikely that a set of correctly matched offers should have exhibited such big variations in prices. While it is possible that this is simply a reflection of different pricing strategies by the merchants, we think a more likely explanation is that the offers are fundamentally different in some ways, for example, TVs sold with accessories versus just the TVs, or the price is inclusive of shipping or extended warranty. We believe a representative price history should avoid mixing these different offers, as the underlying product sold is different. Instead, a better solution is to group the like offers together and present separate price histories for each of the groups.

Despite both being based on the same probabilistic model, Latent performs significantly worse than Match. Examining some of the products for which the difference is large, we notice that poor performance often happens when either (1) the composition of the set of offers changes, causing a large discrete jump occurs in the true price history, or (2) there is an offer considered likely to be correct by the model exhibits a temporary change in prices. As the latent price variables attempt to balance the price dynamics of the model (as governed by α and σ_α^2) and the set of new observations, Latent enters into a transition period and adjusts to these changes gradually . The errors during this transition period are high. On the other hand, Match reacts immediately under both situations, and therefore does not suffer from a transition period. As both types of situations are common in our dataset, Latent performs worse than Match.

7. CONCLUSION

We study the problem of how to aggregate web offers to create representative price histories, taking into account that offers that are matched to a product are not always correct. To solve this problem, we propose a probabilistic model for how offer prices are generated, extending the classical linear dynamical system to allow for possibly incorrect observations. We give an EM algorithm for finding the parameters that maximize likelihood, and work around the computationally intractable exact inference needed in the E-step by variational approximation. To further speed up the learning

algorithm, we take advantage of the structure of our problem that assumes independence among offers, and establish certain matrix equalities that allow us to circumvent the need to take matrix inverses in computing the Kalman gain filter, leading to quadratic speedup of the algorithm. This technique may be of independent interest.

We conduct an extensive evaluation of our approach using data obtained from a commerce search engine. We consider two metrics in evaluation—MASE for evaluating how well a method tracks the true price level, and MACD for evaluating how well a method tracks the true price trend. Our method significantly outperforms alternatives including the median, the trimmed mean, and post-processing using Kalman filter and smoother. The improvements over the baseline are statistically significant. How the output of the probabilistic model is used has large influence over the performance. We find that it is significantly better to use the learned likelihood of whether offers are correctly matched than to use the latent price variables. This work addresses a limitation of the experiments in [1] which depends on data vendor for product prices, and constitutes a step towards building a system that helps consumers decide when to buy a product.

We conclude with several interesting future research directions. First, can the probabilistic model proposed in the paper be integrated into some matching algorithm to improve the matching process itself? While some proposed matching algorithms in the literature can use price as one of the matching features, we are not aware of any solution that takes advantage of the sequentiality of the offers. Combining a dynamic model of prices like the one presented here with matching will likely be both fruitful and challenging. Second, we have considered only a state-space model with only a single state in our probabilistic model. Better performances may be possible by adopting a higher-dimensional state-space model. One may also want to revisit the generative process for the correctly matched offers, as it is common for merchants to keep offer prices constant for periods of time before adjustments. Finally, looking at the entire set of offers open up new possibility in predicting future product prices. An important direction is to revisit the algorithms proposed in [1], and see if by using all offers one can improve forecasting accuracies, and help make better buy-or-wait recommendations to consumers.

8. REFERENCES

[1] R. Agrawal, S. Ieong, and R. Velu. Ameliorating buyers' remorse. In *Proc. KDD*, 2011.

[2] R. Agrawal, S. Ieong, and R. Velu. Timing when to buy. In *Proc. CIKM*, 2011.

[3] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. *The VLDB Journal*, 18(1):255–276, 2009.

[4] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. Adaptive name matching in information integration. *IEEE Intel. Sys.*, 18(5):16–23, 2003.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal Of The Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[7] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. on Knowl. and Data Eng.*, 19(1):1–16, 2007.

[8] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[9] Z. Gharamani and G. E. Hinton. Parameter estimation for linear dynamical systems. Technical report, University of Toronto, 1996.

[10] Z. Gharamani and G. E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):963–996, 2000.

[11] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, 1993.

[12] P. J. Huber. *Robust Statistics*. Wiley, 1981.

[13] M. Isard and A. Blake. CONDENSATION — conditional density propagation for visual tracking. *Int. Journal of Computer Vision*, 29(1):5–18, 1998.

[14] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[15] K. Kanazawa, D. Koller, and S. Russel. Stochastic simulation algorithms for dynamic probabilistic networks. In *Proc. UAI*, 1995.

[16] A. Kannan, I. E. Givoni, R. Agrawal, and A. Fuxman. Matching unstructured product offers to structured product specifications. In *Proc. KDD*, 2011.

[17] M. Michelson and C. Knoblock. Creating relational data from unstructured and ungrammatical data sources. *Journal of Artificial Intelligence Research*, 31:543–590, 2008.

[18] R. Mitkov. *Anaphora Resolution*. Longman, 2002.

[19] H. B. Newcombe, M. J. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records. *Science*, 130:954–959, October 1959.

[20] P. Ravikumar and W. W. Cohen. A hierarchical graphical model for record linkage. In *UAI*, 2004.

[21] S. Roweis and Z. Ghahramani. A unifying review of linear gaussian models. *Neural Computation*, 1997.

[22] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *KDD*, pages 269–278, 2002.

[23] S. Singh, K. Schultz, and A. McCallum. Bi-directional joint inference for entity resolution and segmentation using imperatively-defined factor graphs. In *ECML-PKDD*, pages 414–429, 2009.

[24] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. Sastry. Kalman filtering with intermittent observations. *IEEE Trans. on Automatic Control*, 49:1453–1464, 2004.

[25] W. E. Winkler. Overview of record linkage and current research directions. Technical report, Bureau of the Census, 2006.

[26] P. Zarchan and H. Musoff. *Fundamentals of Kalman Filtering: A Practical Approach*. AIAA, 2nd ed., 2005.

[27] J. Zhong and S. Sclaroff. Segmenting foreground objects from a dynamic textured background via a robust kalman filter. In *Proc. ICCV*, 2003.