The BLMP: Constrained Linear Predictors

June 20, 2023

This document is a neatened up version of various attempts towards deriving the Best Linear Monotonic Predictor.

The Goal

Consider a second order random process X, such that at each value of $t \in \mathbb{R}$, we have a random variable X_t.

We may randomly sample this vector at n points, gaining a vector $\vec{T} = (t_1, t_2, t_3, \ldots)$ of itmes at which the samples were made, and $\vec{X} = (X_{t_i})$. Strictly speaking these are both random variables in and of themselves, up until the moment that we 'realise' them. We can index into these vectors using the the integer $0 \leq i < n$, and we assume without loss of generality that the samples are sorted in time, such that $t_i < t_{i+1} \forall i$.

We wish to find a predictor, $\hat{X}_t$, which will predict the value of X_t at a given value of t, subject to two further conditions:

- The only thing we 'know' (or are willing to *ansatz*) about X_t is the second moment kernel (a generalisation of the covariance):
$$\langle X_t X_s \rangle = k(t, s)$$
- Our predictor should be linear, such that:
$$\hat{X}_t = \vec{a}_t \cdot \vec{X}$$

We again reiterate that $X_t, \vec{X}$ and $\hat{X}_t$ are - strictly speaking - random variables until we make them into real numbers at the moment we wish to actually make a prediction. $\vec{a}_t$ is a real n-tuple, which takes on different values at each value of t.

These are the ingredients of the standard BLP. We extend this by further adding the knowledge that the underlying process – and hence the predictions – should be monotonic in t, such that $X_{t_1} \leq X_{t_2}$ if $t_1 \leq t_2$.

Version 1: The Standard BLP

We first begin by deriving the normal Best Linear Predictor, since it useful to have the derivation as a starting point for our more advanced methods.

Our definition of 'best' will be the standard Mean Squaure Error, averaged across the random variables (and not t, as I was first confused by).

$$\begin{aligned}\mathcal{L} &= \sum_{t\in T} \mathcal{M}_t \\ &= \sum_{t\in T} \langle (X_t - \hat{X}_t)^2 \rangle\end{aligned} \tag{1}$$

Here the sum runs across a set T of chosen 'prediction points' where we wish to estimate the value. However, we note that the problem of predicting X_t is wholly separable; meaning that it is possible to infer the value of each t independently, rather than as a whole. Maximising each $\mathcal{M}_t$ individually will maximise $\mathcal{L}$.

$$\begin{aligned}\mathcal{M}_t(\vec{a}_t) &= \left\langle \left(X_t - \vec{a}\cdot\vec{X} \right)^2 \right\rangle \\ &= \left\langle X_t^2 \right\rangle - 2\cdot\left\langle X_t \vec{a}\cdot\vec{X} \right\rangle + \left\langle (\vec{a}\cdot\vec{X})^2 \right\rangle\end{aligned} \tag{2}$$

Although we have now assembled our cost function in terms of our variable of interest ($\vec{a}_t$), we are in something of a bind , since we know nothing about the behaviour of X_t; not its mean or variance. However, we are willing to conjure up a second moment kernel of some kind, noting that this gives us the following definitions:

$$\left[\vec{k}_t\right]_i = k(t,t_i) = \langle X_t X_{t_i} \rangle \tag{3}$$

$$K_{ij} = k(t_i,t_j) = \left\langle X_{t_i} X_{t_j} \right\rangle \tag{4}$$

Noting that - as a real-valued quantity - $\vec{a}$ passes through the expectation values, we therefore find that Eq. (2) simplifies to:

$$\mathcal{M}_t = \left\langle X_t^2 \right\rangle - 2\vec{k}_t\cdot\vec{a}_t + \vec{a}_t\cdot(K\vec{a}_t) \tag{5}$$

This rearrangement is useful to us as it has rewritten our cost function in terms of a constant-valued (and hence irrelevant) unknown quantity, and functions of the second moment kernel which we are willing to guess at. Computing the derivatives, we find:

$$\frac{\partial \mathcal{M}_t}{\partial \vec{a}_t} = -2\vec{k}_t + 2K\vec{a}_t \quad\Longleftrightarrow\quad \vec{a}_t^{\,\text{best}} = K^{-1}\vec{k}_t \tag{6}$$

Since K is symmetric, K^{-1} is also symmetric, and hence we can write the best linear predictor as:

$$\hat{X}_t = \vec{k}_t\cdot\left(K^{-1}\vec{X}\right) \tag{7}$$

Since K is a function only of $\vec{X}$, the quantity $K^{-1}\vec{X}$ can be precomputed, hence allowing for efficient computation of $\hat{X}$ at all desired values of t.

We also recall that since the problem was totally separable, we could make an inference on a set $t \in T$, and then subsquently expand our predictions to another set $t \in T'$ wihtout recomputing the original predictions.

Version 2: The Peicewise-Local Constraint

We now begin imposing the monotonicity constraints: that $\hat{X}_{t_i} \leq \hat{X}_{t_{i+1}}$, where we have assumed w.l.g. that both $\vec{T}$ and $t \in T$ are sorted to themselves be monotonically increasing. If this is not the case, then the first step is to sort the values such that this is the case.

As the end point of chain, we find that $\hat{X}_{t_0}$ is unconstrained, and hence the Lagrangian is unaltered from the original form of Eq. (5)

$$\hat{X}_{t_0} = \vec{k}_{t_0} \cdot \left(K^{-1} \vec{X} \right) \tag{8}$$

The next steps in the sequence contain a 'slack variable' in the Lagrangian:

$$\mathcal{M}_{t_{i>0}} = \left\langle X_t^2 \right\rangle - 2\vec{k}_t \cdot \vec{a}_t + \vec{a}_t \cdot (K\vec{a}_t) + \lambda \left(\hat{X}_{t_i} - \hat{X}_{t_{i-1}} - s^2 \right) \tag{9}$$

Here $\lambda, s \in \mathbb{R}$, and hence this acts as a constraint enforcing $\hat{X}_{t_i} = \hat{X}_{t_{i-1}} + s^2$, which enforces the monotonicity for all values of s. However, we note that we are not actually interested in the value of s, since this is an *inequality* constraint; we are merely interested in ensuring that $\hat{X}_{t_i} > \hat{X}_{t_{i+1}}$, not controlling *how much greater* it is.

We can therefore make use of the Karush-Kuhn-Tucker (KKT) conditions, and assert that when $s \neq 0$, $\lambda = 0$, and when $\lambda \neq 0, s = 0$: i.e. λ is only non-zero when the constraint is active (i.e. the solution would otherwise break monotonicity).

Optimising this w.r.t. $\vec{a}_t$ (and making the assumption that $\vec{a}_t \neq f(\vec{a}_{t-1})$) gives:

$$\begin{aligned} \frac{\partial \mathcal{M}_t}{\partial \vec{a}_t} &= -2\vec{k}_t + 2K\vec{a}_t + \lambda \vec{X} \\ \vec{a}_t &= K^{-1} \left(\vec{k}_t - \frac{\lambda}{2} \vec{X} \right) \\ \hat{X}_t &= \vec{k}_t \cdot K^{-1} \vec{X} - \frac{\lambda}{2} \vec{X} \cdot K\vec{X} \end{aligned} \tag{10}$$

If the monotonicity constraint is met, and the original BLP solution was monotonic between t_i and t_{i-1}, then the KKT conditions tell us that $\lambda = 0$ and we recover the original solution. If, however, the solution is non-monotonic, then $\lambda \neq 0$, $s = 0$, and we find that:

$$\lambda = 2 \frac{\vec{k}_t \cdot K^{-1} \vec{X} - \hat{X}_{t_{i-1}}}{\vec{X} \cdot K\vec{X}} \tag{11}$$

All of which is to say, that a locally-enforced monotonic (LEM-) BLP takes the form:

$$\hat{X}_{t_i} = \begin{cases} \vec{k}_{t_i} \cdot K^{-1} \vec{X} & \text{if this is } \geq \hat{X}_{t_{i-1}} \\ \hat{X}_{t_{i-1}} & \text{else} \end{cases} \tag{12}$$

In short; whenever the LEM-BLP is identical to the BLO, until the BLP breaks monotonicity, at which point the LEM-BLP predicts a horizontal line, until the BLP rises above this line again.

Version 3: The BLMP

We note that in deriving the LEM-BLP, we made a crucial assumption - namely that we could maintain the separable nature of the solutions, and that minimising $\mathcal{M}_t$ individually would minimise the global Lagrangian $\mathcal{L}$ of Eq. (1). This assumption meant that we could take the derivative of $\mathcal{M}_t$ w.r.t. $\vec{a}_{t_i}$ whilst treating it as a constant w.r.t. $\vec{a}_{t_{i-1}}$.

In a globally-enforced predictor- the Best Linear Monotonic Predictor (BLMP) - this should not be the case; our definition of 'best' should be global, not local.

Hence, in this attempt, we try to minimise $\mathcal{L}$ simultaneously over the entire set of prediction points $t \in T$. This has the important corollary that should we wish to predict a new time $s \notin T$, we would have to re-run the entire global analysis on the set $t \in \{s, T\}$, which might have a drastically different output.

We therefore perform a change of coordinates - rather than trying to predict the random variables $\{\hat{X}_t\}$, we are trying to predict the random variables $\{z_i\}$, which are related to $\hat{X}$ through the following transform:

$$\begin{aligned} \hat{X}_{t_0} &= z_0 \\ \hat{X}_{t_{i>0}} &= \hat{X}_{t_{i-1}} + e^{z_i} \end{aligned} \tag{13}$$

This transformation imposes no constraints on $\hat{X}$ asides from monotonicity, and is therefore totally general on the space of BLMPs. The random variables $\{z_i\}$ are totally unconstrained on $\mathbb{R}$.

For notational convenience, we relabel the states X_{t_i} as X_i, and so the sum $t \in T$ runs over $0 \leq i < N$, and package $\{z_i\}$ into a vector $\vec{z}$. We can then write:

$$\begin{aligned} \hat{X}_i &= S_i(\vec{z}) \\ &= z_0 + \sum_{j=1}^{i} e^{z_j} \end{aligned} \tag{14}$$

The constrained Lagrangian is therefore:

$$\mathcal{L} = \left(\sum_{i=0}^{N-1} \left\langle X_i^2 \right\rangle + \vec{a}_i \cdot K \vec{a}_i - 2\vec{k}_i \cdot \vec{a}_i \right) - \mu_0 \left(\vec{a}_0 \cdot \vec{X} - z_0 \right) - \sum_{i=1}^{N-1} \mu_j \left((\vec{a}_i - \vec{a}_{i-1}) \cdot \vec{X} - e^{z_i} \right) \tag{15}$$

We might wonder why we have left things in terms of $\vec{a}$, when we have already stated our intention to analyse this problem in $\vec{z}$-space. Why go around the houses, when we could just write $\vec{a}_i \cdot \vec{X} = S_i$? Firstly, this is because the dot product is not invertible – and more importantly, it is because we spent quite some time in te BLP derivation writing the Lagrangian in a way that removed our reliance on computing $\langle X_t \rangle$, which we achieved through $\vec{a}$.

Our goal is therefore to find a way to express $\vec{a}_i$ in terms of S_i, such that we can rewrite our Lagrangian in terms of functions only of $\vec{z}$.

The Lagrangian is optimised at a value of $\vec{a}_i$:

$$\begin{aligned}\vec{a}_i &= K_i^{-1}\left(\vec{k}_i - \frac{\Delta_i}{2}\vec{X}\right)\\ \Delta_i &= \begin{cases}\mu_{i+1} - \mu_i & \text{if } 0 \leq j < N-1\\ -\mu_{N-1} & \text{else}\end{cases}\end{aligned} \tag{16}$$

We then write:

$$\vec{a}_i = \vec{v}_i - \frac{\Delta_i}{2}\vec{w} \tag{17}$$

$$\vec{v}_i = K^{-1}\vec{k}_i \tag{18}$$

$$\vec{w} = K^{-1}\vec{X} \tag{19}$$

Since $\vec{a}_i \cdot \vec{X} = S_i$, by definition, we therefore have:

$$\begin{aligned}S_i &= A_i - \frac{\Delta_i}{2}B\\ A_i &= \vec{v}_i \cdot \vec{X}\\ B &= \vec{w} \cdot \vec{X}\end{aligned} \tag{20}$$

We note that A_i is simply the BLP predictor, and B (like $\vec{w}$) is a quantity which depends only on $\vec{X}$. Therefore, we find that:

$$\vec{a}_i = \vec{v}_i + \frac{S_i - A_i}{B}\vec{w} \tag{21}$$

Putting this back into the Lagrangian, we find that our constraint terms vanish since they are automatically satisfied by our reparameterisation, and:

$$\begin{aligned}\mathcal{L}(\vec{z}) &= \left(\sum_{i=0}^{N-1}\left\langle X_i^2\right\rangle + \vec{a}_i(\vec{z}) \cdot K\vec{a}_i(\vec{z}) - 2\vec{k}_i \cdot \vec{a}_i(\vec{z})\right)\\ &\vdots\\ &= \sum_{i=0}^{N-1}\text{constant in terms of } \vec{z} + \frac{1}{B}\left(S_i^2 - S_i\left(A_i + \vec{k}_i \cdot \vec{w}\right)\right)\\ &= \sum_{i=0}^{N-1}\text{constant in terms of } \vec{z} + \frac{1}{B}\left(S_i^2 - 2S_iA_i\right)\end{aligned} \tag{22}$$

Where the last equality follows from the fact that $A_i = \vec{X} \cdot \left(K^{-1}\vec{k}_i\right) = \vec{k}_i \cdot \left(K^{-1}\vec{X}\right) = \vec{w} \cdot \vec{k}_i$, due to the symmetry of K. Hence, the derivative with respect to z_j is:

$$\frac{\partial\mathcal{L}}{\partial z_j} = \frac{2}{B}\sum_i (S_i - A_i) \times \frac{\partial S_i}{\partial z_j} \tag{23}$$

Recalling the definition of S_i, we find:

$$\begin{aligned}\frac{\partial S_i}{\partial z_j} &= \frac{\partial}{\partial z_j}\left(z_0 + \sum_{k=1}^{i} e^{z_k}\right) \\ &= \begin{cases} 1 & \text{if } j = 0 \\ e^{z_j} & \text{if } 0 < j \leq i \\ 0 & \text{else} \end{cases}\end{aligned} \tag{24}$$

Hence:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial z_0} &= \frac{2}{B}\sum_{j=0}^{N-1}(S_j - A_j) \\ \frac{\partial \mathcal{L}}{\partial z_{i>0}} &= \frac{2}{B}e^{z_i}\sum_{j=i}^{N-1}(S_j - A_j)\end{aligned} \tag{25}$$

We note that the gradient is obviously zero if $A_j = S_j$, and we note that A_j is identically equal to the BLP prediction. This therefore demonstrates that in the cases where the BLP is already monotonic, the BLMP predictions will be identical.

If the BLP is not identical, then it is impossible to set $S_j = A_j$, as S_j is constrained to be monotonically increasing in t_j, and so the solutions differ.

Version 4: Shifted BLMPs

One of the known quirks of a BLP is that for small kernel smoothing distances (i.e. a small 'memory'), the prediction will revert to a value of 0

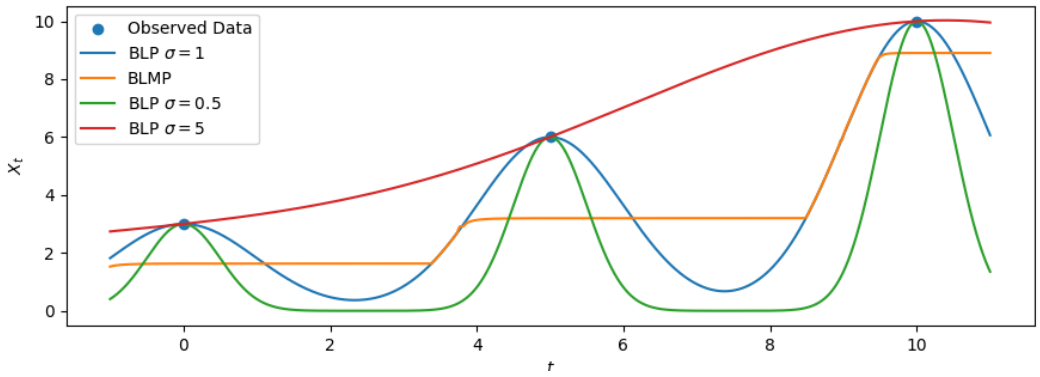

Figure 1: A plot of the toy problem and the resulting BLP and BLMP predictions

The toy problem

Consider the simple dataset $t = \{0, 5, 10\}$, $\vec{X}_t = (3, 6, 10)$, and suppose a Gaussian Kernel with a smoothing length of 1. The standard BLP for this problem would take the following form:

$$K = \sigma^2 \begin{pmatrix} 1 & e^{-12.5} & e^{-50} \\ e^{-12.5} & 1 & e^{-12.5} \\ e^{-50} & e^{-12.5} & 1 \end{pmatrix} \tag{26}$$

$$K^{-1} = \frac{1}{\sigma^2(1 - e^{-25})^2} \begin{pmatrix} \frac{1}{e^{-25}+1} & -e^{-25/2} & \frac{1}{1+e^{25}} \\ -e^{-25/2} & 1 + e^{-50} & -e^{-25/2} \\ \frac{1}{1+e^{25}} & -e^{-25/2} & \frac{1}{e^{-25}+1} \end{pmatrix} \tag{27}$$

$$\vec{k}_t = \sigma^2 \begin{pmatrix} \exp\left(-\frac{t^2}{2}\right) \\ \exp\left(-\frac{(t-5)^2}{2}\right) \\ \exp\left(-\frac{(t-10)^2}{2}\right) \end{pmatrix} \tag{28}$$

$$\hat{X}_t = 3\exp\left(-\frac{t^2}{2}\right) + 6\exp\left(-\frac{(t-5)^2}{2}\right) + 10\exp\left(-\frac{(t-10)^2}{2}\right) + O(10^{-6}) \tag{29}$$

A plot of this predictor (and its BLMP equivalent) is shown in Fig. 1. We can see that when the prediction points are a significant way away from the sampled points, then the data reverts back to a value of zero – a state of affairs which is obviously worsened when the smoothing distance is shortened. The BLMP struggles to cope with the resulting up-and-down predictions, and tries to strike a middle course which places its predictions nowhere near the datapoints themselves.

We might naturally say that this was a fault of choosing a smoothing length which was far too small for our data – and we can see that when $\sigma = 5$, this problem vanishes and we produce a smooth (monotonic) prediction with no reversion to zero.

However, there might be cases where – due to stochastic noise, perhaps – we do end up with a gap in our data larger than our smoothing distance, in which case we would still end up with a reversion to zero – and

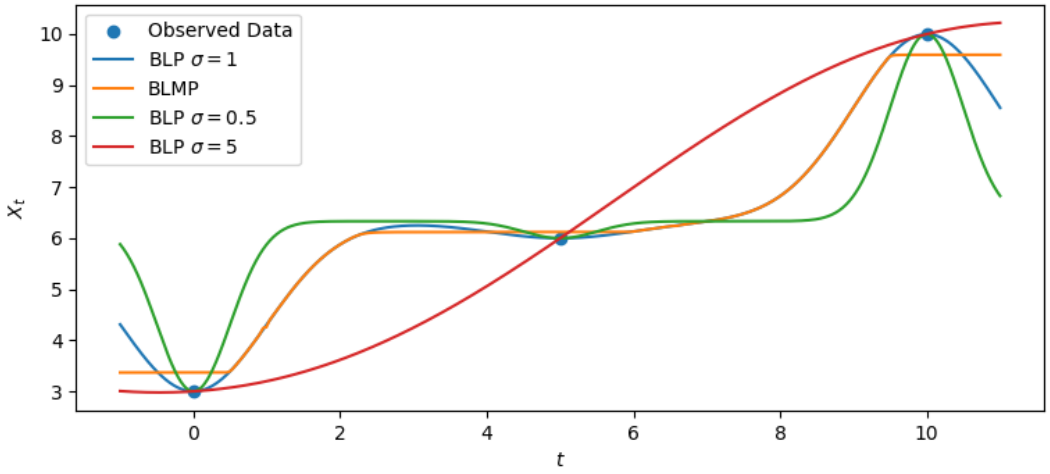

Figure 2: As with Fig. 1, but with mean-reversion.

we might wish to prevent this from happening in some (vaguely) statistically rigorous way.

The Normal Solution

The usual solution that is invoked is to ensure that the data reverts to the *mean*, rather than to zero. This is achieved by performing a transformation on the data, $\vec{X}' = \vec{X} - \mu_X$ (where a vector-scalar subtraction is understodo to occur element-wise), and then computing the BLP on $\vec{X}'$:

$$\hat{X} = \mu_X + \vec{k}_t \cdot K^{-1} \left(\vec{X} - \mu_X \right) \tag{30}$$

Since the mean $\langle X_t \rangle$ is inaccessible (the entire exercise of $\vec{k}$ and K was to sidestep this), it is common to use the sample mean. The result of this is shown in Fig. 2, showing a much more reasonable prediction than before.

However, we note that using the sample mean has its own problems: in Fig 3 we show the same predictor, but evaluated on an extended dataset with samples densly packed around $t = 10$. This biases the mean strongly upwards, and therefore drastically alters the prediction at $t =< 0$, for example, despite no new data being added here.

Formalising the Shifts

As far as I have seen, the subtraction of the mean from the predictor is treated as an 'obvious' thing to do – but this is in fact not quite true as there are several nuances going on. Which are worth discussing – in particular with regards to how they impact a more advanced version of this methodology.

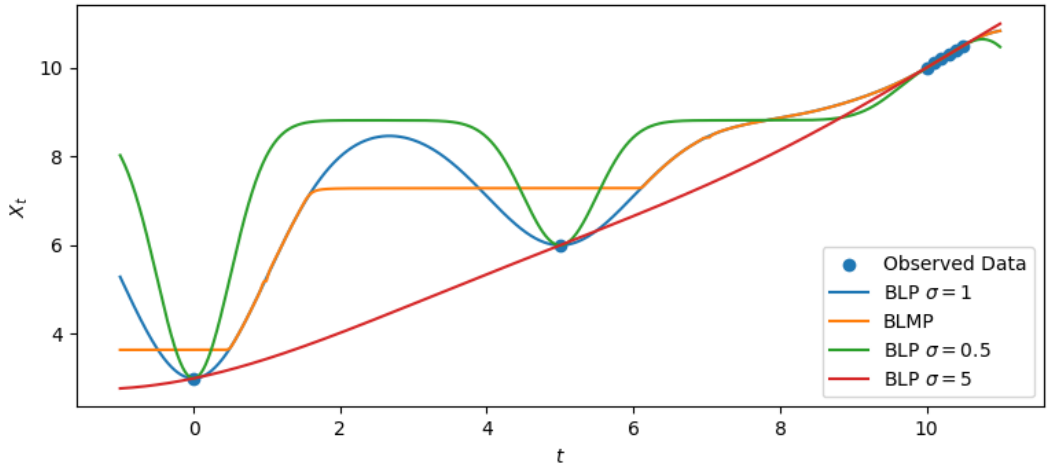

Figure 3: As with Fig. 2, but with additional data added around $t = 10$, showing how altering the mean biases the prediction.

Writing out the analysis in the form of Eq. (2), it is clear that we are making the ansatz that

$$\hat{X}_t = A + \vec{a}_t \cdot \left(\vec{X} - A\mathbb{1}_n\right) \tag{31}$$

Where $\mathbb{1}_n$ is the vector such that $[\mathbb{1}_n]_i = 1 \forall i$, and attempting to minimise the quantity:

$$\begin{aligned} \mathcal{M}_t &= \langle (X_t - \hat{X}_t)^2 \rangle \\ &= \left\langle \left((X_t - A) - \vec{a}_t \cdot \left(\vec{X} - A\mathbb{1}_n\right)\right)^2 \right\rangle \\ &= \left\langle \left(X'_t - \vec{a}_t \cdot \vec{X}' \right)^2 \right\rangle \end{aligned} \tag{32}$$

By comparison, it is clear that the solution is, obviously:

$$\vec{a}_t = K^{-1}\vec{k}_t \iff \hat{X}_t = A + \vec{k}_t \cdot K^{-1}\left(\vec{X} - A\mathbb{1}_n\right) \tag{33}$$

However, an important piece of nuance here is that, in doing this comparison we have changed the nature of $\vec{k}$ and K – in the original analysis they represented the second moments $\langle X_{t_i} X_{t_j} \rangle$, in the mean-shifted analysis they represent the moments between $\left\langle X'_{t_i} X'_{t_j} \right\rangle$. We should therefore denote them as $\vec{k}'$ and K':

$$\left[\vec{k}'_t\right]_i = k'(t, t_i) = \langle (X_t - A)(X_{t_i} - A) \rangle \tag{34}$$

$$K'_{ij} = k'(t_i, t_j) = \left\langle (X_{t_i} - A)(X_{t_j} - A) \right\rangle \tag{35}$$

This is an important distinction because – having imparted some global information into the system – we might be lenient with our Kernel as it no longer has the burden of conveying this global information. I.e. the kernels we might employ differ based on how much information we have already injected.

We also note that it is not possible to try and optimise the BLP with respect to A – i.e., to try and find what value of A will produce the most robust statistical fit. This is obvious upon expanding Eq. (32), since even having redefined $\vec{k}$ and K to include A

$$\begin{aligned} \mathcal{M}_t &= \left\langle X_t^2 \right\rangle - 2A \left\langle X_t \right\rangle + A^2 - 2\vec{k}_t' \cdot \vec{a}_t + \vec{a}_t \cdot K' \vec{a}_t \\ \frac{\partial \mathcal{M}_t}{\partial A} &= -2 \left\langle X_t \right\rangle + 2A - 2\frac{\partial \vec{k}_t'}{\partial A} \cdot \vec{a}_t + \vec{a}_t \cdot \frac{\partial K'}{\partial A} \vec{a}_t \end{aligned} \tag{36}$$

Computing the derivatives of $\vec{k}'$ and K w.r.t. A gives:

$$\begin{aligned} \frac{\partial \vec{k}'}{\partial A} &= (2A - \langle X_t \rangle)\mathbb{1}_n - \left\langle \vec{X} \right\rangle \\ \frac{\partial K_{ij}}{\partial A} &= 2A - \langle X_i \rangle - \langle X_j \rangle \end{aligned} \tag{37}$$

Hence:

$$\frac{\partial \mathcal{M}_t}{\partial A} = -2 \left\langle X_t \right\rangle + 2A - 2(2A - \langle X_t \rangle)\vec{a}_t \cdot \mathbb{1}_n + 2\vec{a}_t \cdot \left\langle \vec{X} \right\rangle + 2 \left(A\vec{a}_t \cdot \vec{a}_t - 2(\vec{a}_t \cdot \left\langle \vec{X} \right\rangle)(\vec{a}_t \cdot \mathbb{1}_n) \right) \tag{38}$$

This clearly has non-cancelling terms in both $\langle X_t \rangle$ and $\left\langle \vec{X} \right\rangle$, both inaccessible terms and hence the derivative is non-computable and an optimal value of A cannot be found.

This reinforces the idea that although performing a BLP on a transformed dataset is valid, no particular shift is any 'more valid' than any other from a rigour standpoint, the validity of the transformed BLP (t-BLP) relies on the following assumptions:

- I have had a suitable transform handed down to me by an external higher power, and I have good reason to believe it will improve the quality of my predictions
- I am happy to invoke my kernel as a relationship between my transformed parameters, rather than the observed ones

The Non-Trivial Corollary

With this in hand, we can now make something of a non-trivial leap – noting that due to the separable nature of the BLP, the value of A does not have to be a constant across all values. We may repeat the above analysis with $A = g(t)$, with the transform being replaced by:

$$\vec{X}' = \vec{X} - \vec{G} \iff [\vec{G}]_i = g(t_i) \tag{39}$$

In this case – as long as we are once again happy to assert $g(t)$ as an *a priori* function, as well as asserting our kernel as a relationship on $(X_{t_i} - g(t_i))$ – we once again recover the t-BLP as being:

$$\hat{X}_t = g(t) + \vec{k}_t \cdot K^{-1} \left(\vec{X} - \vec{G} \right) \tag{40}$$

We can see that this trivially reduces into the previous case when $g(t) = A$. Fig. 4 shows an example of this in action, using the line which joins the end data points ($y = 0.7x + 3$) as the transform.

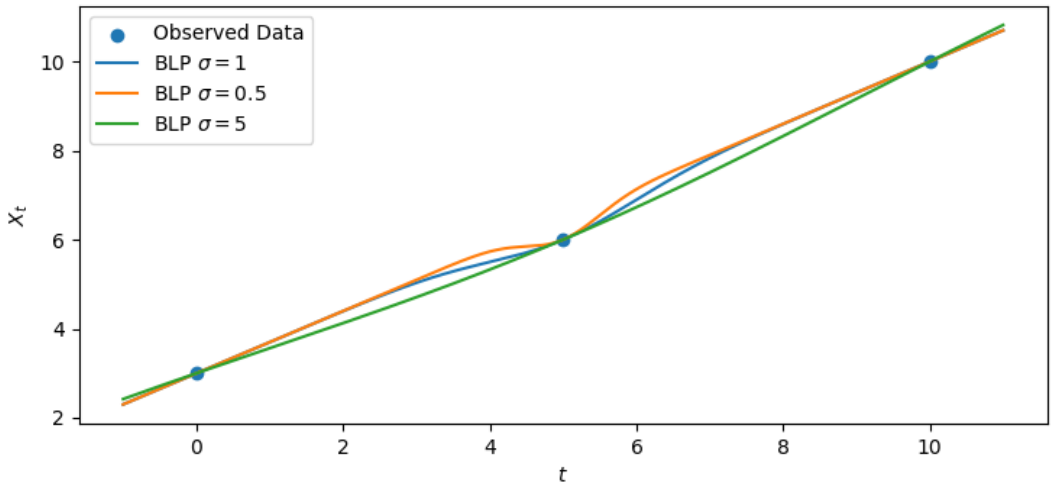

Figure 4: As with Fig. 2, but with a linear transform ($g(t) = 0.7t + 3$) applied.

Even with the smallest smoothing distance, we see that the prediction now reverts to a much more linear picture. Of course from a conceptual standpoint this is not a particularly attractive thing to have done – one of the advantages of the BLP approach is that it is non-parametric, and we have essentially just added a layer of curve fitting on top.

However, we may think of the transform as essentially a prior on the functional form, which the observed data then updates. If we have sufficient data, then the predictor will ignore the prior – as demonstrated in Fig. 5 where the prior is (intentionally) nothing like the true relationship. Aside from a few gaps in the data where the $\sigma = 0.5$ attempts to revert to the prior $g(t) = 0.7t + 3$, we see that the predictors are happy to ignore the prior in favour of the data, as we should expect.

In interpreting the transform as a prior on the predictor, we see that it is therefore inappropriate for $g(t)$ to be determined based on the data – the prior should be the knowledge of the predictor *before* the data is gathered. Even in the simple (and very commonly used) case where $g(t) = \frac{1}{n}\vec{X} \cdot \mathbb{1}_n$, i.e. the sample mean, this is 'bootstrapping' the prior, which may lead to faulty conclusions.

Transforms on the Constrained Predictor

We now turn to the case of formalising the result of a constrained BLP –i.e. the BLMP – using such a 'Prior Transform'. This has an additional level of complexity because our constraint is on the untransformed data (i.e. $X_t \geq X_s$ if $t > s$), whilst the optimisation happens on the transformed data. We shall investigate if this results in any major discrepancies.

Using the same encodings as before ($\hat{X}_{t_i} = f_i(\vec{z})$), the Lagrangian becomes:

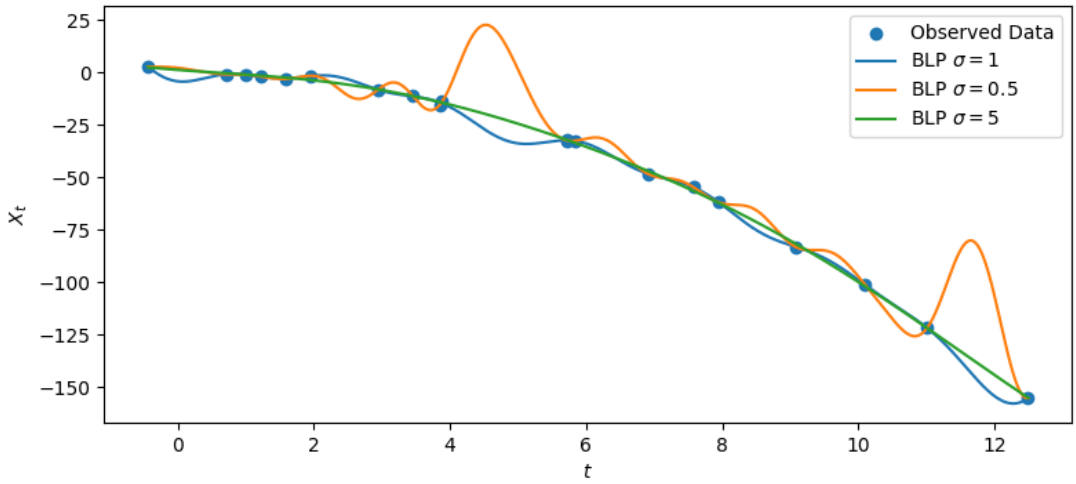

Figure 5: As with Fig. 4, but where the data follows $X(t) = -t^2$+noise, showing that the predictor can ignore the prior (which remains at $g(t) = 0.7t + 3$) in the face of overwhelming data

$$\begin{aligned}\mathcal{L} = &\left(\sum_{i=0}^{N-1}\left\langle {X'}_i^2\right\rangle + \vec{a}_i \cdot K'\vec{a}_i - 2\vec{k}'_i \cdot \vec{a}_i\right) - \mu_0\left(\vec{a}_0 \cdot \vec{X} + g(t_0) - z_0\right) \\ &- \sum_{i=1}^{N-1} \mu_j \left((\vec{a}_i - \vec{a}_{i-1}) \cdot \vec{X}' - e^{z_i} + g(t_i) - g(t_{i-1}))\right)\end{aligned} \tag{41}$$

Where again we have used $X'_t = X_t - g(t)$, and K' and $\vec{k}'$ are the second moment kernels on this transformed space. The optimal value of $\vec{a}_t$ in the t-BLMP are therefore identical to the solutions of the BLMP, with the addition of some primes: We then write:

$$\vec{a}'_i = \vec{v}'_i - \frac{\Delta_i}{2}\vec{w}' \tag{42}$$

$$\vec{v}'_i = {K'}^{-1}\vec{k}'_i \tag{43}$$

$$\vec{w}' = {K'}^{-1}\vec{X}' \tag{44}$$

Since $\vec{a}_i \cdot \vec{X} = S_i - g(t_i)$, by definition, we therefore have:

$$\begin{aligned}S_i &= A'_i - \frac{\Delta_i}{2}B' + g(t_i) \\ A'_i &= \vec{v}_i \cdot \vec{X}' \\ B' &= \vec{w}' \cdot \vec{X}'\end{aligned} \tag{45}$$

Therefore, we find that:

$$\vec{a}'_i = \vec{v}'_i + \frac{S_i - A'_i - g(t_i)}{B'}\vec{w}' \tag{46}$$

And hence:

$$\mathcal{L}(\vec{z}) = \text{const} + \sum_i S_i(S_i - 2(A'_i + g(t_i))) \tag{47}$$

Hence:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial z_0} &= \frac{2}{B}\sum_{j=0}^{N-1}(S_j - (A_j + g(t_j))) \\ \frac{\partial \mathcal{L}}{\partial z_{i>0}} &= \frac{2}{B}e^{z_i}\sum_{j=i}^{N-1}(S_j - (A_j + g(t_j))) \end{aligned} \tag{48}$$

Version 5: A General Formulation

We note that the formulation of Eq. (47) is seems like it might be general, insofar as it does not (yet) contain any constraints - it is simply a reparameterisation of the Lagrangian in terms of the prediction values (S_i), the kernel and the sampeled data. As we shall see, it's not *quite* general – but almost.

It is worth recalling why this is useful - since it might appear that we have gone a roundabout way of computing things, given that the prediction values ($\hat{X}_t$) appeared in the *original* formulation, before we even introduced the kernel.

The reason that the standard BLP formalism uses $\vec{a}_t$, K and $\vec{k}_t$ is because they allow us to rewrite the Lagrangian in terms of these known terms, whilst relegating the uncomputable terms (namely $\langle X_t \rangle$) into a constant term. Since Lagrangians are invariant under the addition of total derivatives (and therefore constant terms), this has no impact on finding the optimal value of $\vec{a}$ – we therefore neatly sidestep the inaccesible terms, by being willing to make an *ansatz* about the form of K and – in the transformed version – $g(t)$.

This methodology therefore makes $\mathcal{L}$ computable - however it is now written in terms of $\vec{a}_t$, instead of $\hat{X}_t$. If we wish to impose conditions on $\hat{X}_t$, this is obviously less than desirable – so we might wish to transform *back* into $\hat{X}_t$ space (which we write as $S_i = \hat{X}_{t_i}$ for convenience). If $\vec{a}_t$ were a scalar, this would be a simple matter of writing $\vec{a}_i = S_i/X_i$, but the dot product is non-invertible.

The entire rigmarole above, therefore, amounts to finding a way to rewrite $\vec{a}_t$ as a function of S_i, which then allows us to write $\mathcal{L}$ as a function only of S_i, K and $\vec{k}_t$. This can then be optimised over $\{S_i\}$, hence finding a general optimum – of course, if S_i is unconstrained then we recover $S_i = A_i + g(t_i)$, the standard BLP solution.

A General Proof

The above statement makes the claim that Eq. (47) is a general result - however, the formulation of the Lagrangian is explicitly that of the monotonic predictor. We therefore produce a general result, which should make this evident...

We begin, as before, with the standard BLP Lagrangian, with a sum over i assumed to index into the N prediction points $t_i \in T$:

$$\mathcal{L}(\{\vec{a}_i\}|\vec{X}, k) = \left(\sum_{i=0}^{N-1} \left\langle {X'}_i^2 \right\rangle + \vec{a}_i \cdot K' \vec{a}_i - 2\vec{k}'_i \cdot \vec{a}_i\right) - \sum_j \lambda_j h_j(\{S\}) \tag{49}$$

Here $h_j(\{S\})$ is the j^{th} constraint on the final prediction values ($S_i = \hat{X}_{t_i} = g(t_i) + \vec{a}_i \cdot \vec{X}'$), such that $h_j(\{S\}) = 0$ if and only if the constraint is obeyed. We make the additional restriction that h_j must be a *linear* constraint, such that it can be written in the form:

$$h_j = A - \sum_k b_{jk} S_k \tag{50}$$

Under this assumption, we can clearly see that:

$$\begin{aligned} \frac{\partial h_j}{\partial \vec{a}_i} &= \sum_m \frac{\partial h_j}{\partial S_m} \frac{\partial S_m}{\partial \vec{a}_i} \\ &= -b_{ji} \vec{X}' \end{aligned} \tag{51}$$

Therefore,

$$\begin{aligned} \frac{\partial L}{\partial \vec{a}_i} &= 2K'\vec{a}_i - 2\vec{k}'_i + \underbrace{\left(\sum_j \lambda_j b_{ji}\right)}_{\eta_i} \vec{X}' \\ &= 2K'\vec{a}_i - 2\vec{k}'_i + \eta_i \vec{X}' \end{aligned} \tag{52}$$

Since η_i is a function only of the Lagrange multipliers and the (constant) constraint terms, it is a constant in terms of $\vec{a}_i$, and so:

$$\begin{aligned} \frac{\partial L}{\partial \vec{a}_i} = 0 \iff \vec{a}_i &= {K'}^{-1}\left(\vec{k}' - \frac{\eta_i}{2}\vec{X}'\right) \\ &= \vec{v}'_i - \frac{\eta_i}{2}\vec{w}' \end{aligned} \tag{53}$$

We can then impose our constraint that $S_i = g(t_i) + \vec{a}_i \cdot \vec{X}$ to recover:

$$\vec{a}_i = \vec{v}'_i + \frac{S_i - A'_i - g(t_i)}{B'} \tag{54}$$

This is the exact result we had before, but written in the case of a much more general linear constraint.

We therefore have:

$$\mathcal{L} = f(X_t, \vec{X}|k, T) + \frac{1}{B'} \sum_i S_i \left(S_i - 2\left[A'_i + g(t_i)\right]\right) - \sum_j \lambda_j h_j(\{S\}) \tag{55}$$

We now make the assumption that $\{S\}$ is parameterised through another vector, $\vec{z}$ such that:

$$\begin{aligned} S_i &= \mathcal{T}_i(\vec{z}) \\ h_j\left(\{\mathcal{T}_i(\vec{z})\}\right) &= 0 \quad \forall\ j, i \end{aligned} \tag{56}$$

I.e., $\mathcal{T}$ is a transform which guarantees all of the constraints are met, meaning that $\vec{z}$ is an unconstrained vector. The Lagrangian can now be written in terms of this unconstrained vector, giving the function to be optimised as (after some optimum-preserving rescaling):

$$\mathcal{L}'(\vec{z}|\mathcal{T}, \vec{X}, k) = \sum_i \mathcal{T}_i(\vec{z}) \left(\mathcal{T}_i(\vec{z}) - 2\left[\vec{X}' \cdot {K'}^{-1}\vec{k}'_i + g(t_i)\right]\right) \tag{57}$$

The derivative of this function is:

$$\begin{aligned} \frac{\partial \mathcal{L}'}{\partial z_m}(\vec{z}) &= \sum_i 2\left(\mathcal{T}_i(\vec{z}) - \left[\vec{X}' \cdot {K'}^{-1}\vec{k}'_i + g(t_i)\right]\right) \left.\frac{\partial \mathcal{T}_i}{\partial z_m}\right|_{\vec{z}} \\ &= 2\sum_i \left(\mathcal{T}_i(\vec{z}) - Q_i\right) \left.\frac{\partial \mathcal{T}_i}{\partial z_m}\right|_{\vec{z}} \end{aligned} \tag{58}$$

Some Examples

The Trivial Case: BLP

The simplest case the unconstrained case, with a suitable transform being $\mathcal{T}_i(\vec{z}) = z_i$. The derivative therefore simplifies to:

$$\begin{aligned} \frac{\partial \mathcal{L}'}{\partial z_m}(\vec{z}) &= \sum_i 2\left(z_i - Q_i\right)\delta_{im} \\ &= z_m - Q_i \end{aligned} \tag{59}$$

Hence, the optimum lies at:

$$z_m = g(t_m) + \vec{k}'_m \cdot \left({K'}^{-1}\left(\vec{X} - \vec{G}\right)\right) \tag{60}$$

This is precisely the BLP case.

The Monotonic Case: BLMP

In the monotonic case, we write our transform as:

$$\mathcal{T}_i(\vec{z}) = \begin{cases} z_0 & i = 0 \\ \mathcal{T}_{i-1}(\vec{z}) + e^{z_i} & \text{else} \end{cases} \tag{61}$$

Which gives the derivative:

$$\frac{\partial \mathcal{T}_i}{\partial z_m} = \begin{cases} 1 & \text{if } m = 0 \\ e^{z_m} & \text{if } 0 < m \leq i \\ 0 & \text{else} \end{cases} \tag{62}$$

This recovers the result we saw above.

The Normed Case: BLNP

A new case we might consider is the case were $\sum_i w_i S_i = C$, as might be the case if our function was constrained to have a certain integral value. In this case w_i act as the weights of each abscissa, for which many algorithms exist – we stick with a naive case of $w_i = \Delta t = \text{const}$, and write $C' = C/\Delta t$.

In this case, a suitable transform would be:

$$\mathcal{T}_i(\vec{z}) = \begin{cases} \frac{C'}{1+\sum_m e^{z_m}} & i = 0 \\ \frac{C' e^{z_i}}{1+\sum_m e^{z_m}} & \text{else} \end{cases} \tag{63}$$

Note that the dimensionality of the problem has been reduced - $\vec{z}$ has dimensions $N - 1$, since this is an equality constraint. This has derivatives

$$\frac{\partial \mathcal{T}_i}{\partial z_m} = \delta_{im} \mathcal{T}_i - \frac{1}{C'} \mathcal{T}_m \mathcal{T}_i \tag{64}$$

Therefore the Lagrangian derivative is:

$$\begin{aligned} \frac{\partial \mathcal{L}'}{\partial z_m}(\vec{z}) &= 2 \sum_i \left(\mathcal{T}_i(\vec{z}) - Q_i\right) \left.\frac{\partial \mathcal{T}_i}{\partial z_m}\right|_{\vec{z}} \\ &= 2\mathcal{T}_m \left[(\mathcal{T}_m - Q_m) - \frac{1}{C'} \sum_i \mathcal{T}_i \left(\mathcal{T}_i - Q_i\right) \right] \end{aligned} \tag{65}$$

The Tightly Bounded Predictor: BLBP

Consider now the case where the predictor values are tightly bounded, such that $C \leq S_i \leq D \forall i$. A suitable encoding might be:

$$\mathcal{T}_i(\vec{z}) = \frac{D + Ce^{-z_i}}{1 + e^{-z_i}} \tag{66}$$

Hence:

$$\begin{aligned} \frac{\partial \mathcal{T}_i}{\partial z_m} &= (D - C) \frac{e^{-z_m}}{(1 + e^{-z_m})^2} \delta_{im} \\ \frac{\partial L}{\partial z_m} &= 2(D - C) \left(\mathcal{T}_m - Q_m\right) \frac{(\mathcal{T}_m - C_m)(D - \mathcal{T}_m)}{D^2} \end{aligned} \tag{67}$$

Perhaps logically, we see once again that this has solutions:

$$\mathcal{T}_m = \begin{cases} Q_m & \text{if } C \leq Q_m \leq D \\ C & \text{if } Q_m < C \\ D & \text{if } Q_m > D \end{cases} \tag{68}$$

Recalling that $\mathcal{T}_m = Q_m$ is the BLP solution, we see that the Best-Linear-Bounded-Predictor simplifies trivially to the BLP subject to a floor-and-ceiling truncation.

The Minimum Gradient Case: BLMGP

Suppose that we know that a predictor should everywhere have a minimum gradient G, such that:

$$\hat{X}_t \geq \hat{X}_s + G(t-s) \tag{69}$$

This uncoding is slightly unusual insofar as it is a function of t as well as $\hat{X}_t$, but since the set T must be determined before the optimisation, this is not particularly a problem.

We therefore generate a vector $\vec{\Delta}$, such that $\Delta_i = t_i - t_{i-1}$. The encoding therefore is simply:

$$\mathcal{T}_i(\vec{z}) = \begin{cases} z_0 & i = 0 \\ \mathcal{T}_{i-1}(\vec{z}) + G\Delta_i + e^{z_i} & \text{else} \end{cases} \tag{70}$$

Note that if $G = 0$ then we have recovered our monotonic predictor. Since g and Δ_i are constants, the derivative is the same as the BLMP: Which gives the derivative:

$$\frac{\partial \mathcal{T}_i}{\partial z_m} = \begin{cases} 1 & \text{if } m = 0 \\ e^{z_m} & \text{if } 0 < m \leq i \\ 0 & \text{else} \end{cases} \tag{71}$$

0.1 The Smooth Positive Case: BLSPP

A notable feature of some of the other predictors is that, when they reach the edge of their constraints, a hard boundary is formed – a predictor which is constrained to $\hat{X}_t \geq 0$ will follow an unconstrained predictor at positive values, only to display a discontinuous derivative when the predictor tries to go negative.

We might therefore attempt to concoct a predictor which is both *positive* and *smooth*.

$$\begin{aligned} \ell_i &= \begin{cases} z_0 & i = 0 \\ \ell_{i-1} + \alpha \left(\frac{e^x - 1}{e^x + 1} \right) & \text{else} \end{cases} \\ \mathcal{T}_i(\vec{z}) &= \exp(\ell_i) \end{aligned} \tag{72}$$

The exponential term in $\mathcal{T}_i$ ensures that the predictor is always positive, whilst the α term controls how different ℓ_i and ℓ_{i-1} are allowed to be, forcing the bound such that for $\alpha > 0$:

$$e^{-\alpha}\mathcal{T}_{i-1} \leq \mathcal{T}_i \leq e^{\alpha}\mathcal{T}_{i-1} \tag{73}$$

This then enforces a smoothness condition. The derivative of this transform is:

$$\frac{\partial \mathcal{T}_i}{\partial z_m} = \begin{cases} \mathcal{T}_i & \text{if } m = 0 \\ 2\alpha \mathcal{T}_i \frac{e^{z_m}}{(1+e^{z_m})^2} & 0 < m \leq i \\ 0 & \text{else} \end{cases} \tag{74}$$