EE194/290S: ML for Sequential Decision Making **Fall 2018**

Lecture 13: Adaptation 3

Lecturer: Vidya Muthukumar *Scribes: A Domingos, M Wiggert, S Mataraso*

13.1 Adapting the Model Size

13.1.1 Prediction based on memory length

Consider some binary sequence with either 0 or 1 representing the value at a certain time-point, for example: $y_1, y_2, \ldots, y_n = (0, 1, 0, 1, 0, 1, 0, 1, \ldots)$. Our current best performing algorithm, AdaHedge, achieves a bounded regret (in comparison to one single action in hindsight) of order $\sqrt{T}$, on this sequence. But we intuitively understand that after seeing this alternating sequence many times we should be able to anticipate the alternating nature of the sequence, and reduce or eliminate our losses. To do that we can define a memory-dependent functions f, where $f : y_{t-1} \to y_t$ and $f : \{0, 1\} \to \{0, 1\} \in \mathbf{F}$, where $\mathbf{F}_{t-1} = \{0, 1\}$ is the result at the last time step. For the next time step, $\mathbf{F}_t$, we have the following potential models to choose from:

$$\mathbf{F}_t : \left\{ \begin{array}{cc} \begin{bmatrix} 0 \to 0 \\ 1 \to 0 \end{bmatrix}_A, & \begin{bmatrix} 0 \to 1 \\ 1 \to 0 \end{bmatrix}_B \\ \\ \begin{bmatrix} 0 \to 0 \\ 1 \to 1 \end{bmatrix}_C, & \begin{bmatrix} 0 \to 1 \\ 1 \to 1 \end{bmatrix}_D \end{array} \right\}$$

In the context of the previously introduced sequence, the best function choice is the function in row 1, column 2 (subscript B), which depending on the output for the last time step (t-1) will predict the opposite for the current time step (t). This model would result in accurately predicting the alternating sequence.

13.1.2 Prediction based on memory length D

For a fixed memory length D, we can build the model class $f : y_{t-D}, \ldots, y_{t-1} \to y_t$ where $f : \{0, 1\}^D \to \{0, 1\} \in \mathbf{F}_D$. The number of models in the function class $\mathbf{F}_D$ increases dramatically with D. For binary possible outcomes $\{0, 1\}$ there are 2^D different sequences/contexts $\{0, 1\}^D$. Because each model f maps each context to $\{0, 1\}$, there are $2^{\#contexts}$ possible functions, meaning the size of the model class for memory length D is:

$$|\mathbf{F}_D| = 2^{2^D}$$

A problem that comes with the exponentially growing size of the model class is overfitting on sequences that are of low-length or no temporal structure e.g. Ber(0.7). This can be observed on Figure 13.1. For the FTL algorithm it takes only very few rounds to determine a clear leader (one or zero), from that point on-wards it collects all rewards and doesn't accumulate any more regret. However, for FTL(3) to determine a leader in each of it the $2^3 = 8$ contexts which then leads to picking one model of the $2^{2^3} = 256$ possible models, it needs to see all of those contexts at least a couple of times. We can see that this happens after around 180 rounds, from which point on-wards the regret is constant. Until that point, when it has learned to predict one in all contexts, it misses out on reward and consequently accumulates less reward then FTL. From that point on-wards its reward curve is parallel to the FTL curve. For higher memory-length models the point when it has found its leader in all contexts happens exponentially later, as the number of contexts grows exponentially with

memory length (e.g. FTL(4) around 500, FTL(5) around 1700), which explains the increasing distance between the reward and regret curves. This is overfitting because although the FTL(4) model works a lot better than FTL on past data (as it remembers all the contexts and what the next number was) but this doesn't generalize, so it makes mistakes on the new data coming in.

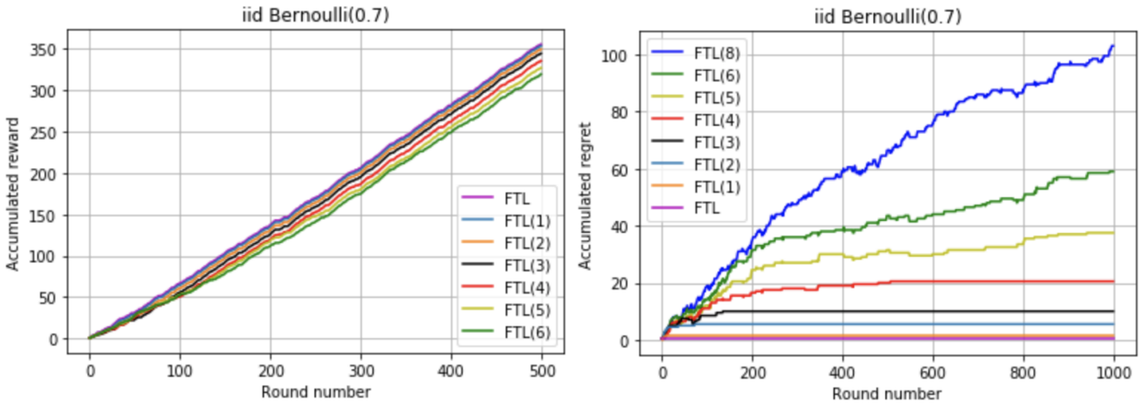

Figure 13.1: A comparison of the accumulated reward on the left and the accumulated regret (compared in hindsight to predictions $\{0\}^T$ or $\{1\}^T$) on the right for different memory length FTLs on an iid Ber(0.7) sequence. As the sequence is not memory dependent, the higher the memory length of the model the longer it takes to learn the best model (always one) in all of the exponentially increasing number of contexts and consequently the worse the rewards during this learning time. Correspondingly, the accumulated regret is higher the higher the memory length of the model.

13.2 Model Selection in Offline Learning

13.2.1 Empirical Risk Minimization

Let's assume we have n data points $Data : z_i = (x_i, y_i)$ i = 1, ... n that are realizations of the random variables X and Y, representing the covariates and their respective responses. Further we assume that there exists a joint probability distribution $P(x, y)$ and our n data points are i.i.d. samples from that distribution (remark: we don't have to believe that there is something like a true distribution but still act as if there would be one).

We want to pick the best predictor function $f : X \to Y$ from a function class F (e.g. polynomials up to degree d). Using the terminology of empirical risk minimization we can define the risk associated with a specific function $f \in F$ as $\overline{L}(f) = \mathbb{E}[l(Y, f(X))]$ where $l(y, f(x))$ is a loss function.

Take for example polynomial regression with $y = f_d(x)$ where $f_d \in F_d$, F_d being all polynomials of degree d. The best predictor function in this function class is the one that minimizes the risk, $f^*(d) = \arg\min_{f \in F_d} \overline{L}(f)$. We can't calculate $\overline{L}(f)$ directly and therefore can't get $f^*(d)$. But we can empirically estimate it from our n iid data points:

$$\hat{f}_n = \arg\min_{f \in F_d} \frac{1}{n} \sum_{i=1}^{n} l(y_i, f(x_i))$$

According to the law of large numbers $\hat{f}_n \approx f^*(d)$ for $n \to \infty$. Consequently, we can assume that $\overline{L}(\hat{f}_n) \approx \overline{L}(f^*(d))$. Indeed for polynomial regression we can establish a bound for the expected estimation error $\mathbb{E}[\hat{f}_n - f^*(d)] \leq \mathcal{O}(\frac{d}{n})$. If we want this error to be smaller than ϵ, this means we need $n >> n_0(d)$ where $n_0 \propto \frac{d}{\epsilon}$.

How to choose d given n samples?
1) minimize the risk estimation error $E_{estimation} = \overline{L}(\hat{f}_n) - \overline{L}(f^*(d))$ which decreases when d decreases.

2) minimize the risk approximation error $E_{approximation} = \overline{L}(f^*(d))$ which decreases when d increases.

In general terms we can bound the estimation error $E_{estimation} = \overline{L}(\hat{f}_n) - \overline{L}(f^*(d)) \leq \gamma_d(n)$ where $\gamma_d(n)$ is some function of n that generally increases for $d \uparrow$ and $n \downarrow$.

13.2.2 Model Selecting Empirical Risk Minimization

Knowing the estimation approximation error trade-off we want to prevent overfitting which happens when we have complex models and too few data points. Taking $\gamma_d(n)$ as the estimation error term which is dependent on n and the complexity of the function $order(f)$, we want to choose the model complexity appropriately to the number of datapoints n.

With $f \in F_D$ where D is the maximal model order and a measure of the model complexity $order(f)$ we can use the estimation error term $\gamma_{order(f)}(n)$ to explicitly penalize more complex models.

$$\hat{f}_n = \arg\min_{f \in F_D} \Big[\frac{1}{n} \sum_{i=1}^{n} l(y_i, f(x_i)) + \gamma_{order(f)}(n) \Big]$$

The first term, an estimate for the approximation error, is bounded by the range of the loss function. As $\gamma_{order(f)}(n)$ is decreasing as n increases this means that with higher n we'll pick more complex models.

13.3 Model Selection in Online Learning

13.3.1 Model Selecting FTL

Consider a formulation of follow the leader where the model has some memory of length D. The loss function is defined at time t for some function $f \in \mathbf{F_D}$ as:

$$L_{t-1,f} = \sum_{s=D}^{t-1} \mathbf{1}[f(Y_{s-D}, ...Y_{s-1}) \neq Y_s]$$

The best model to predict the next value in the sequence would be determined by the model that minimizes our loss so far:

$$\hat{f}(t) = \arg\min_{f \in \mathbf{F_D}} L_{t-1,f}$$

However, it is clear that there is a major limitation to the model above, namely the model order must be pre-specified. Follow the leader can be modified in order to select not just the best model of a given order but the best model regardless of order.

If we use the minimize the loss function and remove the restriction of model order, thus allowing models of any order, the chosen model will be overfit. This is because the minimization problem always has a solution with minimum 0 by setting the model equal to order $t-1$ and selecting the model that matches the realized values in the sequence.

The current loss term, $L_{t-1,f}$ refers to the fit of the model to the observed data points. In terms of the approximation-estimation error trade off, minimizing this term will give a very low approximation error and high estimation error. In order to avoid the problem of overfitting, we can add a term to the optimization problem that represents estimation error and is some function of the order of the class of functions we are considering. By modifying the optimization problem to include this term, we penalize higher order models.

We can optimize over a class of models $\mathbf{F_d}$ that contains all models in the set $\{\mathbf{F_0}, \mathbf{F_1}, ...\mathbf{F_D}\}$. Including the above change, our new optimization problem becomes:

$$\hat{f}(t) = \arg\min_{f \in \mathbf{F_d}} (L_{t-1,f} + \ln[order(F_D)])$$

where F_D represents the class to which the chosen model from $\mathbf{F_d}$ belongs. In the follow the leader regime, the order of the model class for a model with memory D is 2^{2^D}, and so our optimization problem becomes

$$\hat{f}(t) = \arg\min_{f \in \mathbf{F_d}} (L_{t-1,f} + \ln(2) * 2^D)$$

We call this regime model selecting follow the leader.

13.3.2 Reward & Regret in Model Selecting FTL

Let's consider a concept of reward for model selecting FTL, defined as:

$$\sum_{t=D}^{T} \mathbf{1}[f(Y_{t-D}, ...Y_{t-1}) == Y_t]$$

Specifically, let's look at reward for FTL with different order models for a 2-periodic sequence:

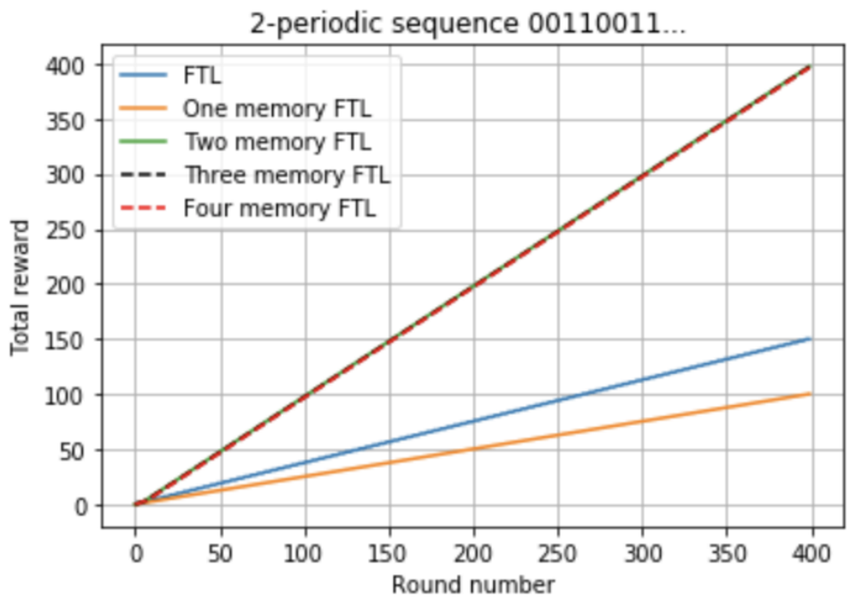

Figure 13.2: Total reward for different FTL model complexities.

We can see that FTL does not perform very well, which is expected. We only accurately predict one of the four values in the sequence, and half the time we correctly guess the next value when the sequence has an equal number of ones and zeroes. We see an even worse performance for one-memory FTL, which is effectively tricked by the 2-repeating nature of the sequence. However, we see two-, three-, and four-memory FTL do well, because they can capture the full complexity of the model. However, we want a way to penalize the increased complexity of the three- and four-memory FTL model over the two-complexity FTL model. To do so, we modify our definition of regret to include a penalty for more complex models:

$$\sum_{t=D}^{T} \mathbf{1}[f(Y_{t-D}, ...Y_{t-1}) == Y_t] - \ln(2) * 2^D$$

where D is the order of the model. Looking at the first 200 rounds with this penalized reward, we see the following:

The three- and four-memory models never achieve the same reward as the two-memory FTL because they have a greater penalty and the predictions from all three are always correct,

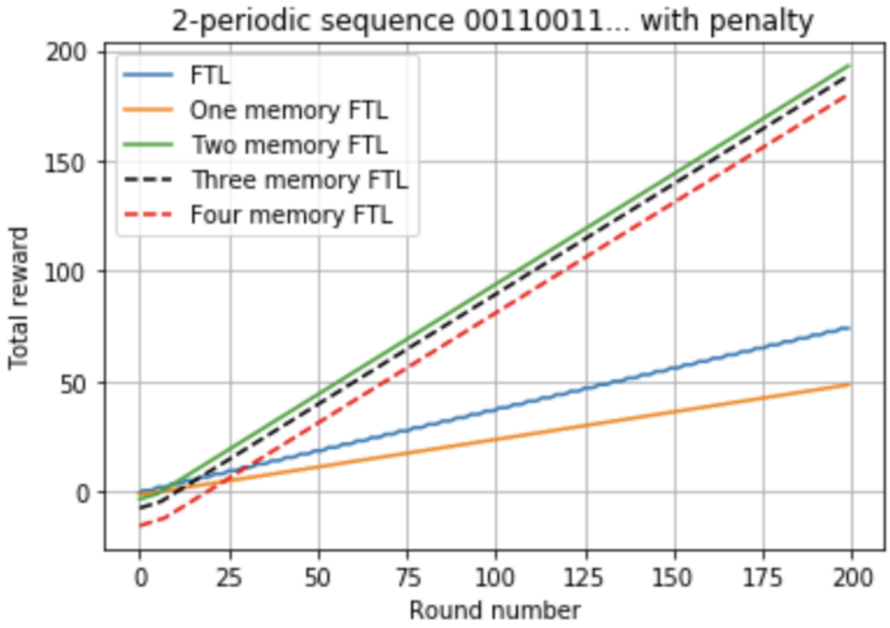

Figure 13.3: Total reward for different FTL model complexities with a penalty for increasing model complexity.

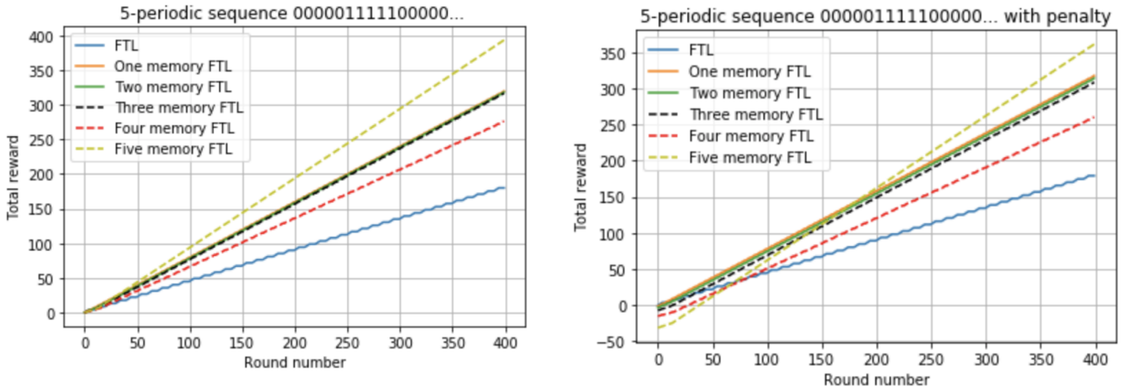

Figure 13.4: Total reward for different FTL model complexities on a 5 periodic sequence. On the left without a penalty and on the right with a penalty for increasing model complexity.

giving the same value of the loss function. Let us consider a more complicated sequence, a 5-periodic sequence.

We can see that only the five-memory FTL algorithm is able to achieve maximum reward. When we add in our penalty, we see similar results, as expected.

The five-memory model originally has less cumulative reward than the other models due to the penalty, but as we collect more data, we see the five-memory FTL model exceeding the performance of the other models. Indeed, whenever the data are generated from a higher order model, more data is necessary to overcome to penalty term down-weighting the use of higher order models. This is similar to the effect that a large amount of data has in overpowering the down-weighting effect of a prior in offline learning, if a prior is used to penalize high complexity models.

Another example is shown in Figure 13.5, where the sequence is stochastic but the parameter of the Bernoulli random variable for the next value in the sequence is determined by the four prior values in the sequence. As expected, we see the reward is maximized for a four-memory FTL model, which can capture the full dependency of the model.

Whenever dealing with the concept of regret, it is important to define a baseline that makes sense in the context of the problem. The maximum order of a model must be appropriately selected. Otherwise, the reference set of models will be too large, resulting in an overfit baseline that isn't realistic to achieve in an online fashion. Without the appropriate selection

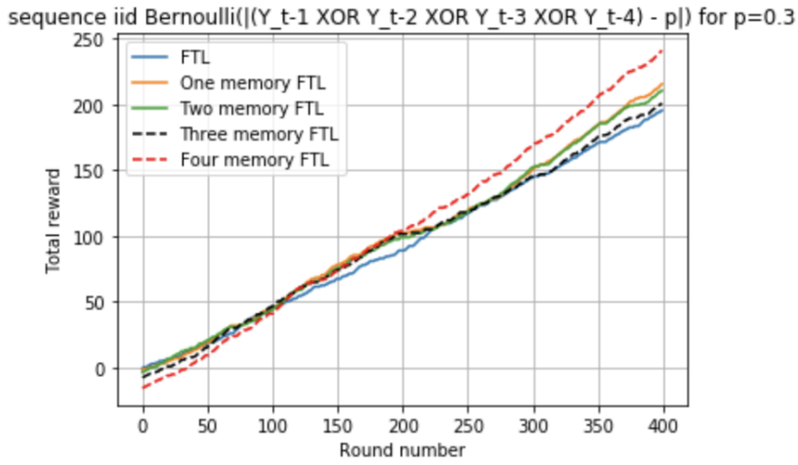

Figure 13.5: Total reward for different FTL model complexities with a penalty for increasing model complexity.

of a baseline, the concept of regret is meaningless for comparison.

13.3.3 Model Selecting Exponential Weights

We can extend the ideas used in model selecting FTL to exponential weights. Recall in the original formulation of exponential weights one modifies the weight at time step t as:

$$w_t = e^{-\eta_t L_{t-1}}$$

In the model selecting version of exponential weights, it is necessary to add a term that penalizes the use of a higher order model:

$$w_{t,f} = e^{-\eta_t L_{t-1,f}} * 2^{-2^{order(f)}}$$

The $2^{-2^{order(f)}}$ term acts as a prior down weighting the probability of selecting a higher order model. However, it is important to note that as t increases, the effect of the prior is washed out by the increasing value of the loss, $L_{t-1,f}$, if an incorrect model is chosen, including one that is of too small of an order. In other words, the increase in penalty for selecting a higher order model is smaller than the decrease in the total loss when the amount of data is large.

13.3.4 Connection to Offline Learning

Clearly, model selection is an important problem in predictive modeling. Without carefully considering the model and its complexity, one is in danger of overfitting the predictive model to the data and building a model which does not generalize. As such, a lot of work has been done in the area of offline model selection. One approach for for offline model selection is minimizing the Akaike Information Criterion (AIC) [1]:

$$AIC = 2k - 2ln(\hat{L})$$

where k represents the number of parameters in the model and $\hat{L}$ is the maximum value of the likelihood function. This model selection criteria bears a similarity to our formulation of empirical risk minimization. Specifically, minimizing AIC balances the trade-off between the penalty for a more complex model and picking a model that is more accurate on seen data. It is important to note that thus far we have been using the simple example of binary sequence prediction and using loss functions. AIC is different in that it requires maximizing the likelihood instead of minimizing the loss; however, the trade-off between a better model and a more complex model remains a key part of the criterion, as in online model selection.

To demonstrate the similar regularizing effect of AIC, consider polynomial regression on a small dataset of $X = \{0,1,2,3,4,5\}$ and $Y_i = X_i + N, N \sim \mathcal{N}(0,0.2)$. Fitting a polynomial

of unconstrained order would result in a 5th degree polynomial to perfectly fits all the data points, or more generally an $n-1$ degree polynomial for n data points. The unconstrained model would look something like Figure 13.6. This model maximizes the likelihood of the data. However, the regularizing effect of the $2k$ term in AIC will prevent overfitting by

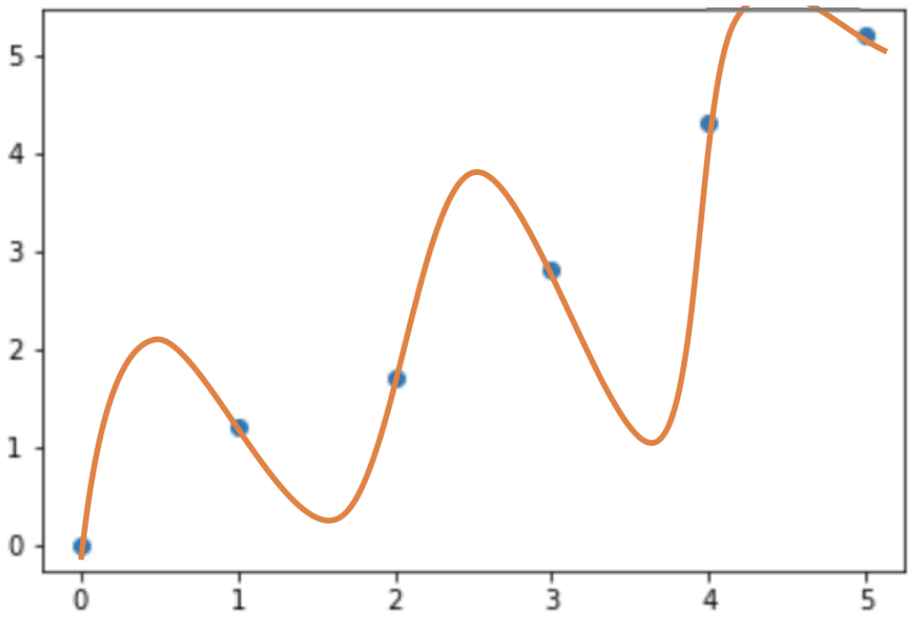

Figure 13.6: Polynomial fit with no complexity penalty term, it picks k=5 a 5th order polynomial.

selecting a model class which is more appropriate. With a linear class of models, the maximum likelihood of the data is less than with a 5th degree polynomial, but still fairly good. However, the reduced complexity (and consequently, reduced model parameters k, meaning reduced complexity penalty) leads the AIC criterion to favor a linear model, with a much more generalizeable fit, visualized below in Fig 13.7. Generally, for a more complex model to

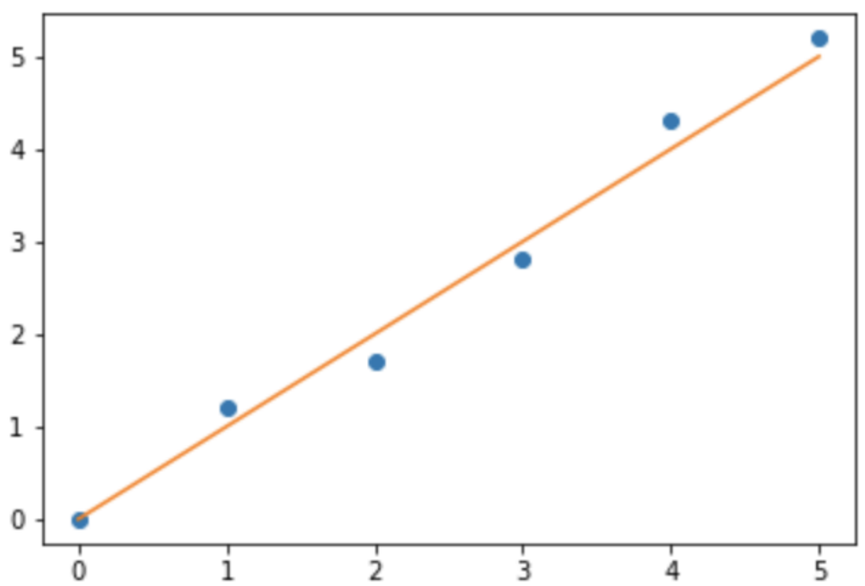

Figure 13.7: Polynomial fit with k=1 complexity penalty term.

be favorable under AIC, the additional complexity must increase the likelihood of the data more than the additional penalty for the added complexity. This is exactly what we observe in the online setting. To pick a more complex model, the additional complexity must result in a great enough decrease in the loss function to overcome the penalty.

13.3.5 Other Considerations in Online Model Selection

As we have seen, many of the regimes we have discussed, such as FTL and exponential weights, can be modified for online model selection by not constraining the model to a certain order. However, it is important to modify the optimization problem or put a prior down-weighting complex models. If this is not done, the resulting models are likely to be overfitting. As with traditional statistical machine learning, the relative importance of the regularizer or prior can be adjusted if information is known about the likely order of the model.

References

[1] Akaike H. Likelihood of a model and information criteria. Journal of econometrics. 1981;16(1):3-14.