Time Series 3

-Quantifying the World-

Lecturer: Darren Homrighausen, PhD

Time Series Decomposition

Components

General Model:

$$Y_t = T_t + S_t + \epsilon_t$$

(Note that there is the multiplicative model, which is this model on the log scale)

Where

- T_t is a trend

  (when there is a long-term change in the data, which doesn't have to be linear)
- S_t is a seasonal component

  (when a series is influenced by seasonal factors, e.g., the quarter of the year, the month, or day of the week. Seasonality is always of a fixed/known period)
- ϵ_t is a stationary component

  (Generally of the form of an ARMA model)

Seasonality

Added to the model via dummy or indicator variables

Example: U.S. employment data tends to vary seasonally

We can adjust for this via regressing Y onto 4 seasonal dummy variables

$$S_t = \beta_1 \mathrm{Summer} + \beta_2 \mathrm{Fall} + \beta_3 \mathrm{Winter} + \beta_4 \mathrm{Spring}$$

After fitting this model, we retain the seasonally adjusted time series

$$\tilde{Y}_t = Y_t - \hat{Y}_t$$

Trend

A basic example of trend estimation is using polynomials

$$T_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \cdots$$

Commonly, only a linear time trend is considered
(higher order polynomials tend to extrapolate poorly)

One we get the fitted values of $\tilde{Y}$ onto T, we can now produce

$$\tilde{\epsilon} = \tilde{Y}_t - \hat{T}_t$$

Now, we can fit an ARMA model to $\tilde{\epsilon}$

Trend

Another example of trend estimation you have seen: moving average

$$\hat{T}_t = \frac{1}{5}(Y_{t-2} + Y_{t-1} + Y_t + Y_{t+1} + Y_{t+2})$$

(Here, for a length 5 window)

Alternatively, we can make a weighted moving average

$$\hat{T}_t = \frac{1}{10}(Y_{t-2} + Y_{t+2}) + \frac{1}{5}(Y_{t-1} + Y_{t+1}) + \frac{2}{5}Y_t$$

What about more general types of local averaging?

Nonparametric regression

Nonparametric regression

Suppose $Y \in \mathbb{R}$ and we are trying to nonparametrically fit the regression function

($Y \in \mathbb{R}$ means Y is a real number)

$$\mathbb{E}Y|X = f_*(X)$$

(Fact: the regression function is the best possible predictor of Y given X. In the time series case, X is commonly time)

A common approach is to specify

- A fixed functions, $(\phi_k)_{k=1}^K$
  (Known as a basis)
- Here K is a tuning parameter

Nonparametric regression

We follow this prescription:

1. Write
$$f_*(X) = \sum_{k=1}^{K} \beta_k \phi_k(X) = \beta^\top \Phi(X)$$
(This is a feature map!)
2. Estimate β with least squares
(Often higher k are rougher $\Rightarrow$ choosing K controls smoothness.)

Example: ϕ_k could be a polynomial transformation of time basis function

Feature map

We start with p covariates

We generate K features

Example: Multiple regression with a feature transformation

$$\begin{aligned}\Phi(X) &= (1, x_1, x_2, \ldots, x_p, x_1^2, x_2^2, \ldots, x_p^2, x_1x_2, \ldots, x_{p-1}x_p) \in \mathbb{R}^K \\ &= (\phi_1(X), \ldots, \phi_K(X))\end{aligned}$$

Before feature map:

$$f_*(X) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j = \beta_0 + \beta^\top X$$

After feature map:

$$f_*(X) = \sum_{k=1}^{K} \beta_k \phi_k(X) = \beta^\top \Phi(X)$$

Feature Map Example

Example: Suppose $X = (\text{income}, \text{height})^\top$

Then we could specify the map

$$\Phi(X)^\top = (1, \text{income}, \text{height}, \text{income} * \text{height}, \text{income}^2, \text{height}^2)$$

So, making polynomial transformations creates a feature map

Many other techniques do as well...

Observation 1: Feature map

For neural networks write:

$$\sigma_k(X) = \sigma\left(\alpha_{k0} + \sum_{j=1}^{p} \alpha_{kj} x_j\right) = \sigma\left(\alpha_{k0} + \alpha_k^\top X\right)$$

Then we have

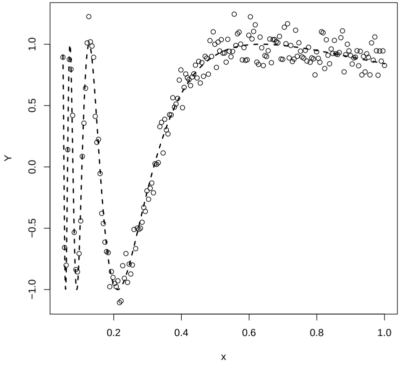
$$\Phi(X) = (1, \sigma_1(X), \ldots, \sigma_K(X)) \in \mathbb{R}^{K+1}$$

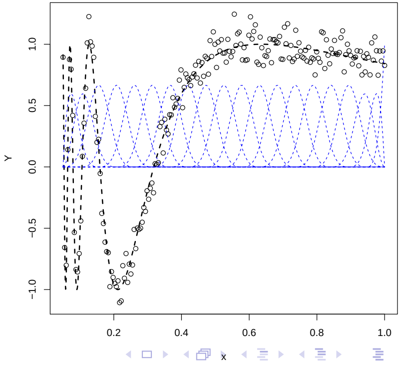
and

$$f_*(X) = \beta^\top \Phi(X) = \beta_0 + \sum_{k=1}^{K} \beta_k \sigma\left(\alpha_{k0} + \sum_{j=1}^{p} \alpha_{kj} x_j\right)$$

Nonparametric regression: Example

```
x = seq(.05,1,length=200)
Y = sin(1/x) + rnorm(100,0,.1)
plot(x,Y)
xTest = seq(.05,1,length=1000)
lines(xTest,sin(1/xTest),col='black',lwd=2,lty=2)
```





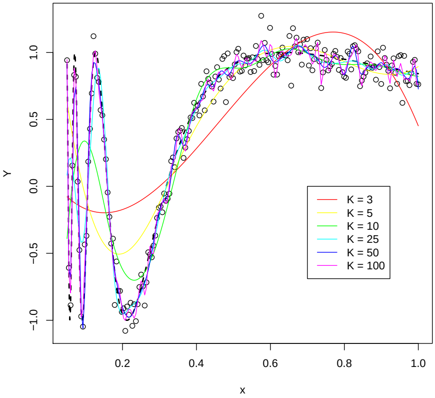
Nonparametric regression: Example

(Code for second plot on previous slide)

```
require(splines)
X = bs(x,df=20)
plot(x,Y)
lines(xTest,sin(1/xTest),col='black',lwd=2,lty=2)
matlines(x=x,X,lty=2,type='l',col='blue')
```

Nonparametric regression: Example

```
require(splines)
X    = bs(x,df=K)
Yhat = predict(lm(Y~.,data=X))
```



Nonparametric regression with Neural networks

We can try to fit it with a single layer NN with different numbers of hidden units/layers K

A notable difference with B-splines is that 'wiggliness' doesn't necessarily increase with K due to regularization

Some specifics:

- I used the R package neuralnet

  (This uses the resilient backpropagation version of the gradient descent)
- I regularized via a stopping criterion ($||\partial\ell||_\infty < 0.01$)
- I did 3 replications

  (This means I did three starting values and then averaged the results)

Neural networks: Example

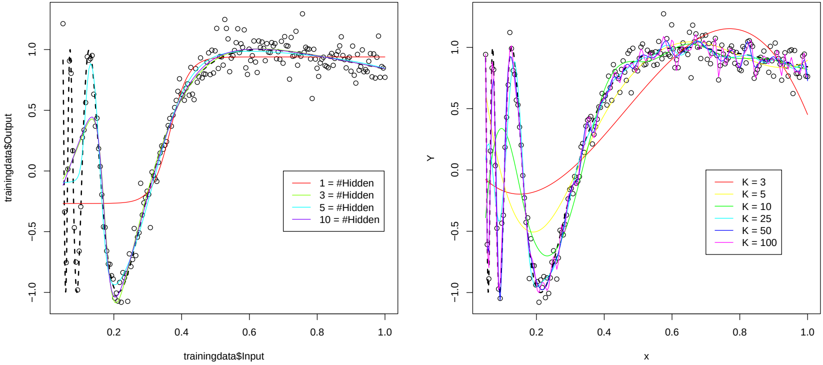


Figure: Single layer NN vs. B-splines

Neural networks: Risk

Let's find the models that minimize the Risk

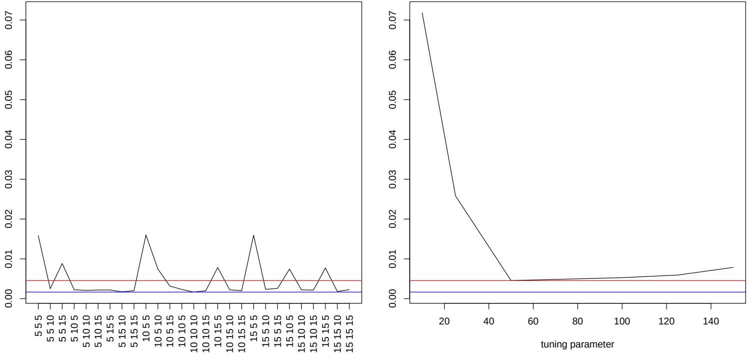


Figure: 3 layer NN[1] (Risk min) vs. B-splines (Risk min)

[1] The numbers mean (#(layer 1) #(layer 2) #(layer 3))

Neural networks: Example

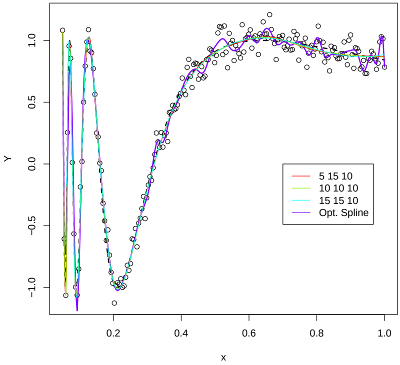


Figure: Optimal NNs vs. Optimal B-spline fit

Neural networks: Code for Example

```
trainingdata  = cbind(x,Y)
colnames(trainingdata) = c("Input","Output")
testdata      = xTest

require("neuralnet")
K          = c(10,5,15)
nRep       = 3
nn.out     = neuralnet(Output~Input,trainingdata,
                       hidden=K, threshold=0.01,
                       rep=nRep)
nn.results = matrix(0,nrow=length(testdata),ncol=nRep)
for(reps in 1:nRep){
  pred.obj = compute(nn.out, testdata,rep=reps)
  nn.results[,reps] = pred.obj$net.result
}
Yhat = apply(nn.results,1,mean)
```

Dickey-Fuller Test

Suppose the model:

$$Y_t = T_t + S_t + \epsilon_t = \beta_0 + \beta t + \phi Y_{t-1} + w_t$$

Hence, the model is a random walk if $\phi = 1$

(A random walk is a particular, though common, type of nonstationary process)

We can rewrite it in terms of differences as

$$Y_t - Y_{t-1} = \beta_0 + \beta t + (\phi - 1) Y_{t-1} + w_t$$

$\rightarrow$ We can test if the regression coefficient on the lag Y_{t-1} is zero

Huge Note: The ADF is a unit-root test (that is, $\phi = 1$). This is only one of many ways a time series can be nonstationary!

(This extends to the Augmented Dickey-Fuller (ADF) via including more AR terms)

Seasonal Trend Loess (STL)

Another, classic nonparametric smoother is called Loess

It combines local averaging with KNN to make a smoothed fit

It also is the centerpiece of a classic time-series paradigm:

Seasonal Trend Loess (STL)

Seasonal Trend Loess (STL)

Reminder:

$$Y_t = T_t + S_t + \epsilon_t$$

- T_t is a trend

  (when there is a long-term change in the data, which doesn't have to be linear)
- S_t is a seasonal component

  (when a series is influenced by seasonal factors, e.g., the quarter of the year, the month, or day of the week. Seasonality is always of a fixed/known period)
- ϵ_t is a stationary component

  (Generally of the form of an ARMA model)

STL iteratively fits loess to the trend/seasonal components to estimate the decomposition

Seasonal Trend Loess (STL)

The two main parameter choices that must be picked are

- Trend window
- Seasonal window

These parameters adjust how quickly the estimators of these components can change

(Smaller values $\leftrightarrow$ quicker change)

Picking a very large seasonal window makes the seasonal fit repeat (hence is called periodic)

Some Comments on Exponential Smoothing

Suppose we want to predict $\hat{Y}_{T+1}$

We could try to use:

- $\hat{Y}_{T+1} = Y_T$

  Or
- $\hat{Y}_{T+1} = \frac{1}{T}\sum_{t=1}^{T} Y_t$

These are opposite extremes

Simple Exponential smoothing uses something between these extremes

$$\hat{Y}_{T+1} = \alpha Y_T + \alpha(1-\alpha) Y_{T-1} + \alpha(1-\alpha)^2 Y_{T-2} + \cdots$$

(The odd form comes from $\hat{Y}_{T+1} = \alpha Y_T + (1-\alpha)\hat{Y}_T$, the α parameter can be found by minimizing the squared residuals.)

Note that simple exponential smoothing is equivalent to an ARIMA(0,1,1) with $\theta = \alpha - 1$