R Tutorial

Part 5

Hidden Markov Model

Its easier to **record the evidence and observations** rather than find the actual reason behind every phenomena

why HMM?!

All we do is recording observation
All we have is observations

depmixS4

- There are different packages in R for building hidden Markov models, such as *mhsmm*, *HMM*, *depmixS4,* and so on.

- Although each of these packages has its own pros and cons, **depmixS4** package is the one we use in this tutorial.

- For more information about this package check the links below:
  https://cran.r-project.org/web/packages/depmixS4/vignettes/depmixS4.pdf
  https://cran.r-project.org/web/packages/depmixS4/depmixS4.pdf

- Install depmixS4 package on your R.

depmixS4

Two steps are involved in using **`depmixS4`** which are illustrated in the following:

1. model specification with function **`depmix`**.
2. model fitting with function **`fit`**.

Maximizing the likelihood of the train data by adjusting the model parameters and fitting the model (Problem 3).

depmixS4

- The Simplest Model (model specifications):

To build a HMM we have to determine two main parameters;
1) The feature (or the features) which the model will be built based on it (or them).
2) Number of states.

```
library("depmixS4")
data()
set.seed(1)

mod <- depmix(response =, data =, nstates = )
```

- `set.seed()`: The seed number is the starting point used in the generation of a sequence of random numbers. Using the same number insures you can reproduce the same results.
- `response`: Basically response is the feature (or the features) that the model is built for (or them).
- `data` should be a `data.frame`.
- `nstates` is the number of states.

The `depmix` function returns an object of class “depmix” which contains the model specification, and not a fitted model. Hence, the model needs to be fitted by calling fit.

depmixS4

- A Simple Model (model fitting):

```
mod <- depmix(response =, data =, nstates = )

fm <- fit(mod)
```

The `fit` function fit the model with the determined specifications, which means over various iterations the log-likelihood will converge at its maximum.

```
> fm <- fit(mod)
iteration 0 logLik: -305.321
iteration 5 logLik: -305.2833
iteration 10 logLik: -305.1962
iteration 15 logLik: -304.8088
iteration 20 logLik: -297.5845
iteration 25 logLik: -91.85991
iteration 30 logLik: -88.73835
iteration 35 logLik: -88.73062
converged at iteration 39 with logLik: -88.73058
>
```

depmixS4

After determining the model's specifications, we can fit the model based on those specifications.

There is two commands which provides more detail on our model, **`print()`** and **`summary()`**.

depmixS4
(Model Specification)

```
iteration 30 logLik: -88.73835
iteration 35 logLik: -88.73062
converged at iteration 39 with logLik: -88.73058
>
> summary(fm)
Initial state probabilties model
pr1 pr2
  0   1

Transition matrix
        toS1  toS2
fromS1 0.884 0.116
fromS2 0.084 0.916

Response parameters
Resp 1 : gaussian
    Re1.(Intercept) Re1.sd
St1           5.510  0.192
St2           6.385  0.244
>
```

Initial State Probability

States Transition Probabilities

Responses Distribution

depmixS4

`depmix` *Dependent Mixture Model Specifiction*

Description

depmix creates an object of class depmix, a dependent mixture model, otherwise known as hidden Markov model. For a short description of the package see depmixS4. See the vignette for an introduction to hidden Markov models and the package.

Usage

```
depmix(response, data=NULL, nstates, transition=~1, family=gaussian(),
prior=~1, initdata=NULL, respstart=NULL, trstart=NULL, instart=NULL,
ntimes=NULL,...)
```

Arguments

response	The response to be modeled; either a formula or a list of formulae (in the multivariate case); this interfaces to the glm and other distributions. See 'Details'.
data	An optional data.frame to interpret the variables in the response and transition arguments.
nstates	The number of states of the model.
transition	A one-sided formula specifying the model for the transitions. See 'Details'.
family	A family argument for the response. This must be a list of family's if the response is multivariate.
prior	A one-sided formula specifying the density for the prior or initial state probabilities.

depmixS4

`initdata`	An optional data.frame to interpret the variables occuring in `prior`. The number of rows of this data.frame must be equal to the number of cases being modeled, `length(ntimes)`. See 'Details'.
`respstart`	Starting values for the parameters of the response models.
`trstart`	Starting values for the parameters of the transition models.
`instart`	Starting values for the parameters of the prior or initial state probability model.
`ntimes`	A vector specifying the lengths of individual, i.e. independent, time series. If not specified, the responses are assumed to form a single time series, i.e. `ntimes=nrow(data)`. If the `data` argument has an attribute `ntimes`, then this is used. The first example in `fit` uses this argument.

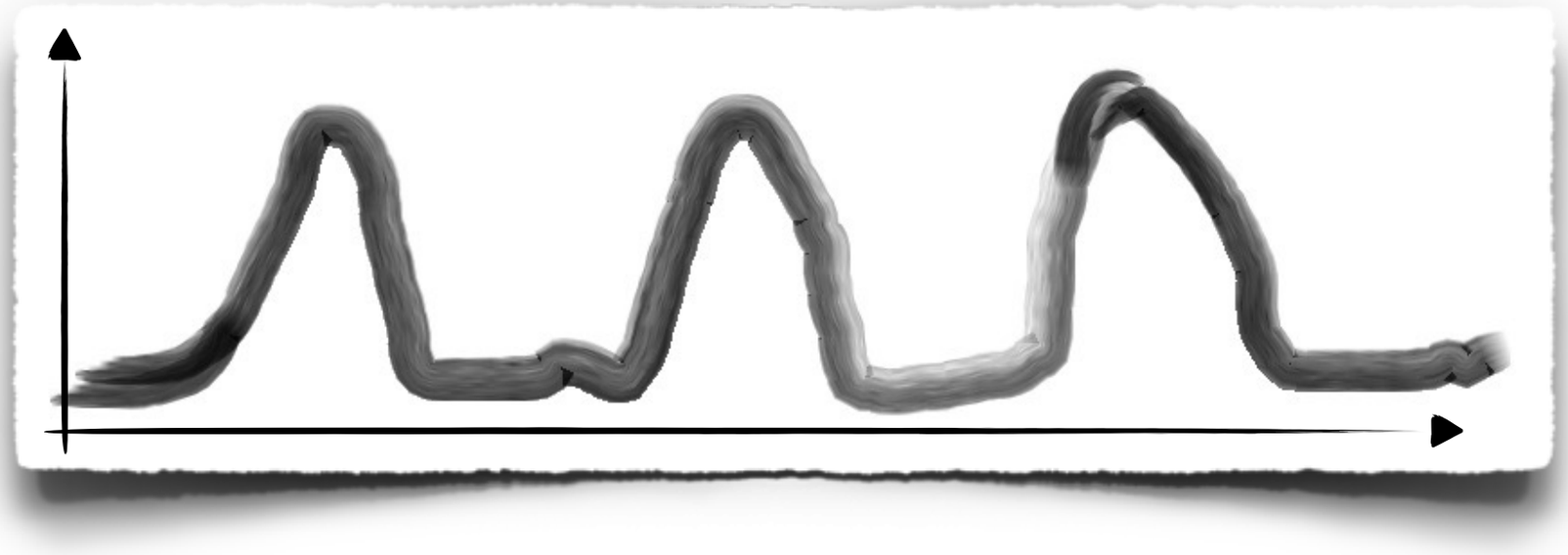
depmixS4

Before going into more details of **`depmixS4`** package, a few concepts and techniques need to be discussed in order to build a better model which represents the training data:

- Determining the number of states (AIC and BIC)
- Overfitting and underfitting
- Feature selection

depmixS4
(Number of States)

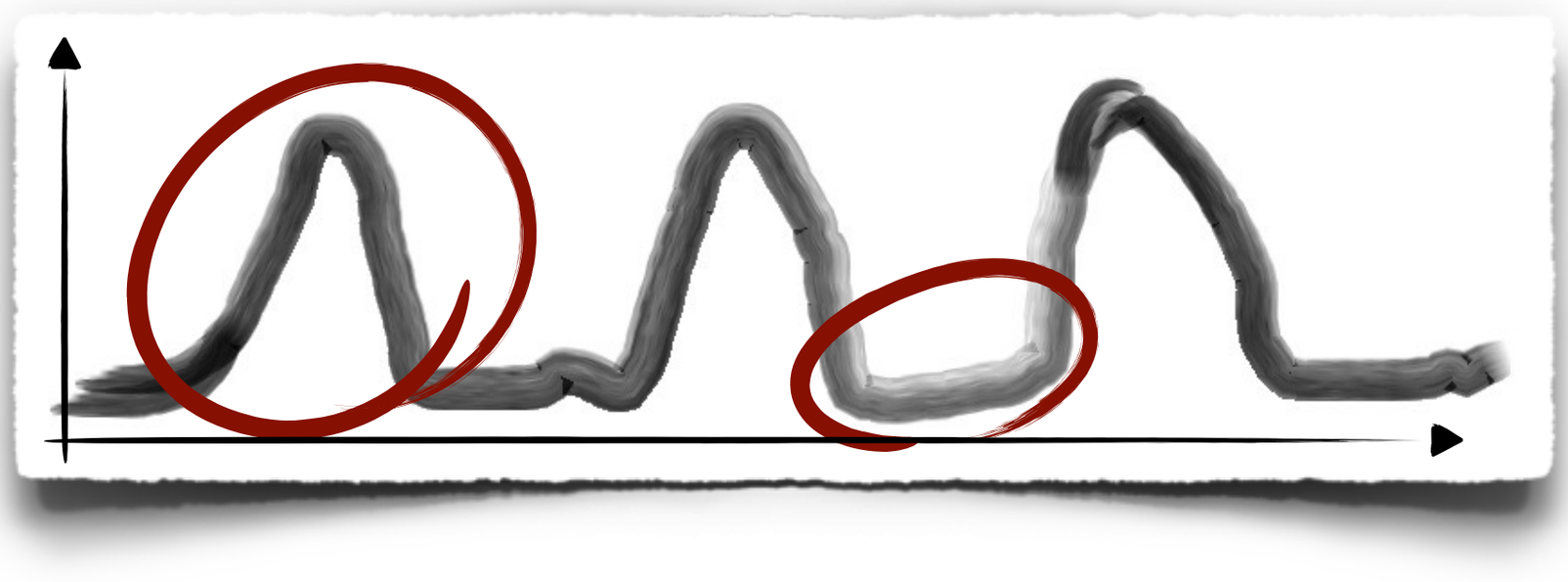
Getting a sense of states in the hidden Markov models by visualizing the observations;



How many states this observation sequence can have?

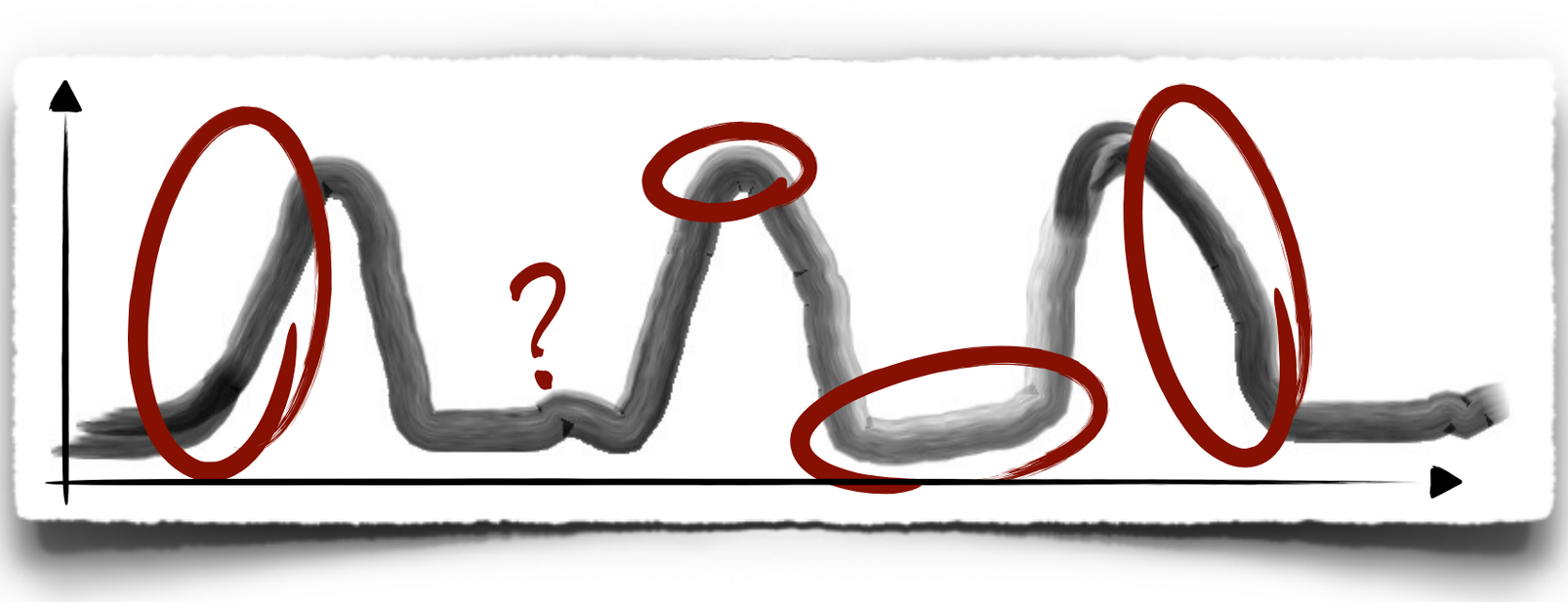
depmixS4
(Number of States)

Getting a sense of states in the hidden Markov models by visualizing the observations;



depmixS4
(Number of States)

Getting a sense of states in the hidden Markov models by visualizing the observations;



By choosing a greater number for states, redundant detail is adding to the model and it may overfit the model.

depmixS4
(Number of States)

How many states are not too much nor too less? What would be the best number of states?

Sometimes by visualizing the training data we can get a rough feeling about the number of states. But this cannot be anything more than a guess for a starting point.

There are few measures which help to avoid overfitting or underfitting when we are adjusting a model's parameters.

depmixS4
(AIC and BIC)

When fitting models, it is possible to increase the likelihood by adding parameters, but doing so may result in overfitting. Both AIC and BIC attempt to resolve this problem.(1)

The model with the lowest BIC is preferred.

Definition [edit]

The BIC is formally defined as[2]

$$\mathrm{BIC} = \ln(n)k - 2\ln(\hat{L}).$$

where

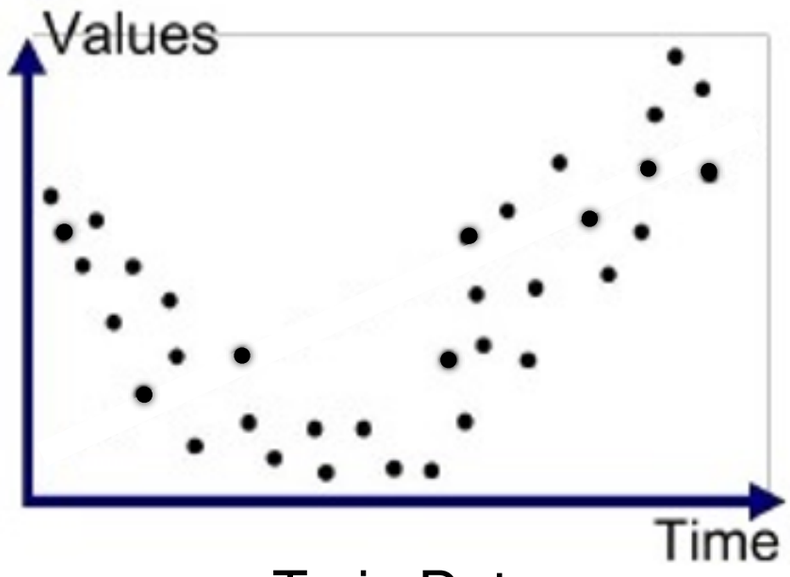
- $\hat{L}$ = the maximized value of the likelihood function of the model M, i.e. $\hat{L} = p(x|\hat{\theta}, M)$, where $\hat{\theta}$ are the parameter values that maximize the likelihood function;
- x = the observed data;
- n = the number of data points in x, the number of observations, or equivalently, the sample size;
- k = the number of parameters estimated by the model. For example, in multiple linear regression, the estimated parameters are the intercept, the q slope parameters, and the constant variance of the errors; thus, $k = q + 2$.

(1) https://en.wikipedia.org/wiki/Bayesian_information_criterion

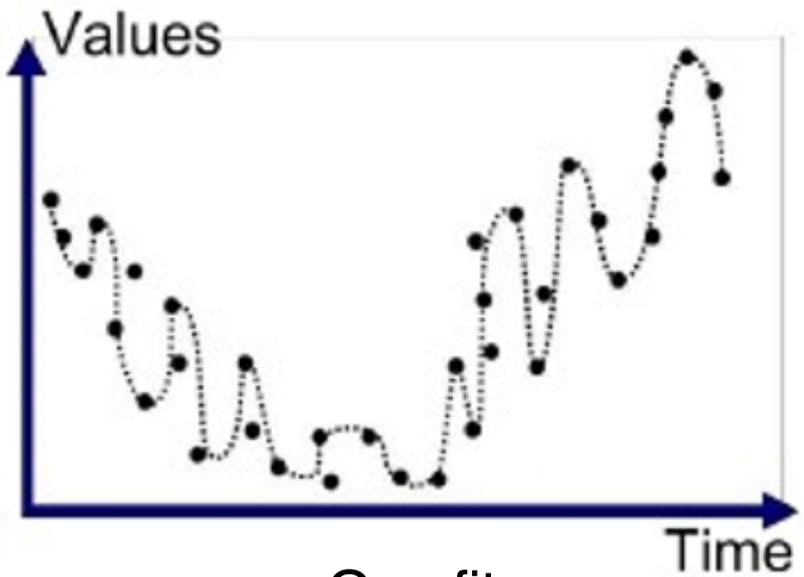
Modeling
(Overfitting and Underfitting)

Overfitting occurs when the model fits the details in the training data to the extent that it negatively impacts the model on test data. This means the normal fluctuations in the training data is picked up by the model.

Overfitting may happen because the parameter of the model is not determine properly.



Train Data

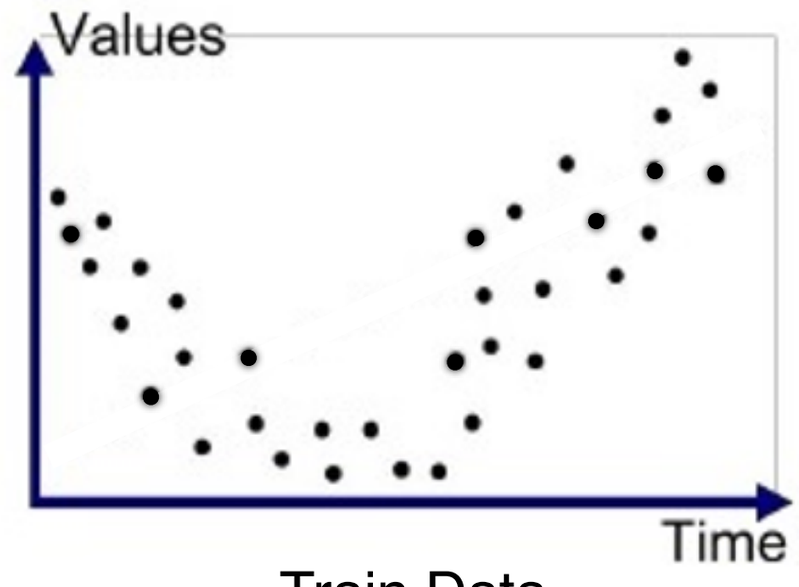


Overfit

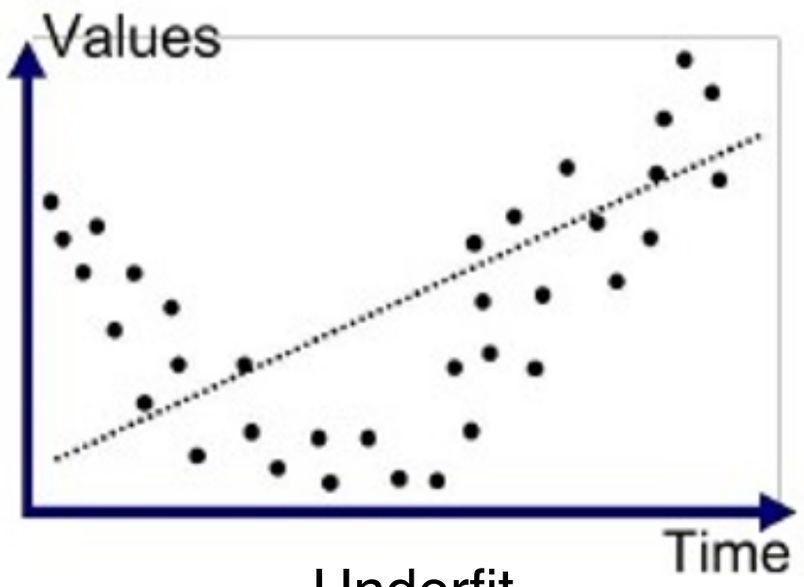
Modeling
(Overfitting and Underfitting)

Underfitting refers to a model that neither fits the training data, nor captures the underlying trend of the training data.

Underfitting may happen because, there was not enough training data, the training data was too general, or the parameter of the model is not determine properly.



Train Data

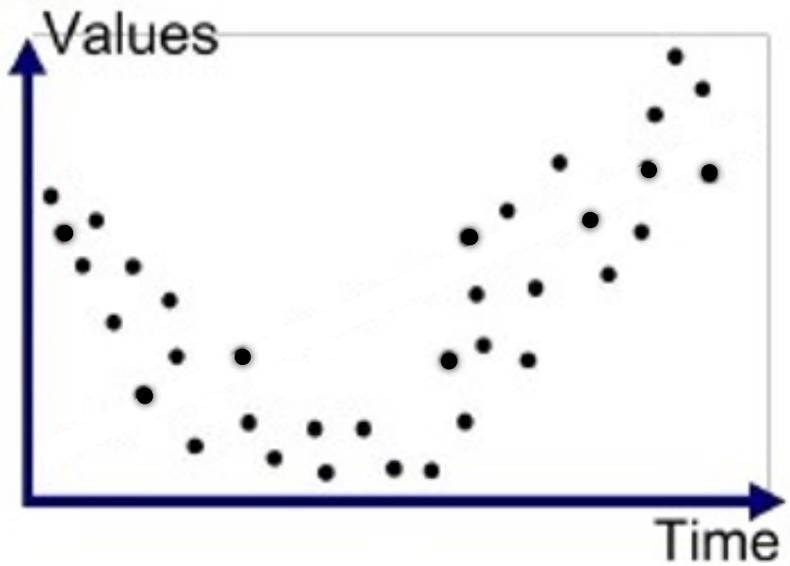


Underfit

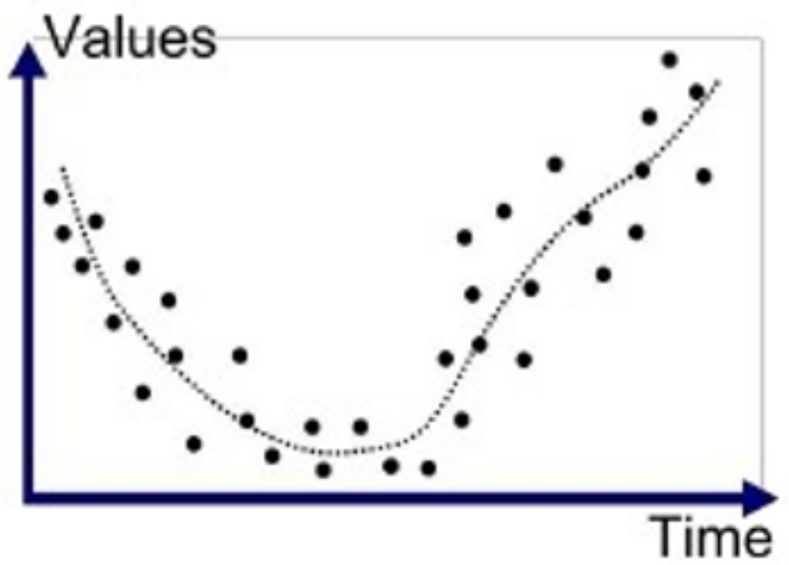
Modeling
(Overfitting and Underfitting)

Ideally, a good model is at the sweet spot between underfitting and overfitting. The goal is to capture the accurate underlying behaviour and characteristics of the train data, however, it is very difficult to do in practice.

Training Assumption: The test data has the similar underlying trend as the training data.



Train Data

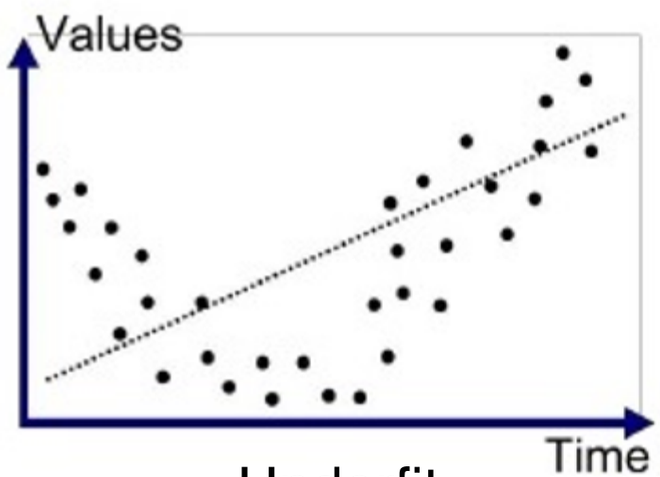


Fit

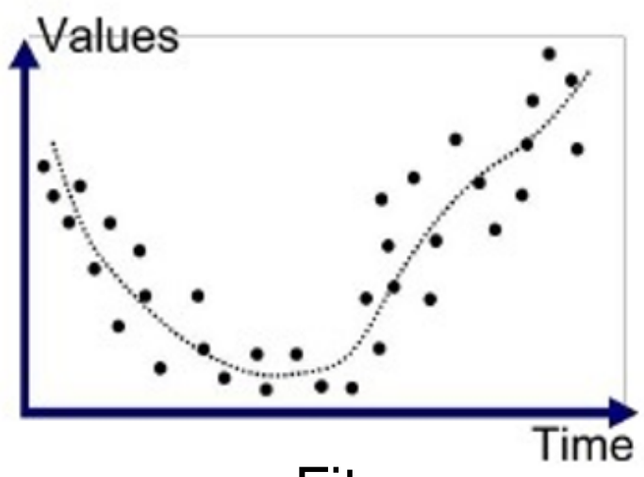
Modeling
(Overfitting and Underfitting)

If we want to evaluate our model based on accuracy;

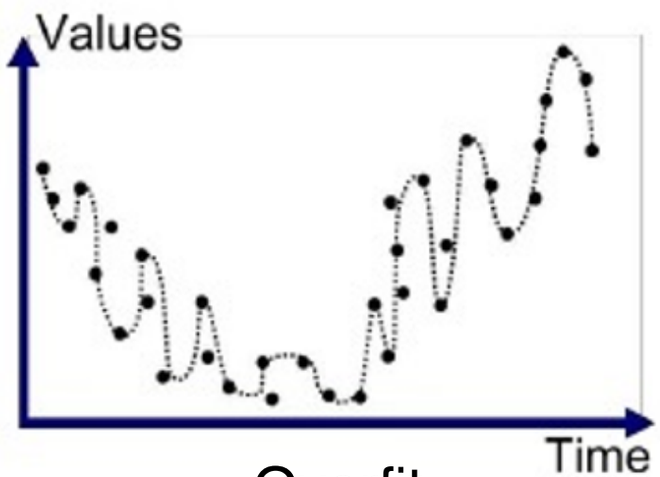
- Overfit model has high accuracy on training data but low accuracy on test data.
- Underfit model has low accuracy on both train and test data.
- Fit model has high accuracy on training and also reasonably high accuracy on test data.



Underfit



Fit



Overfit