Day-14, Oct-16, 2024

#Data Splitting: The simplest way to prevent the model overfitting and underfitting and to choose a model with the right level of complexity is to use data Splitting.

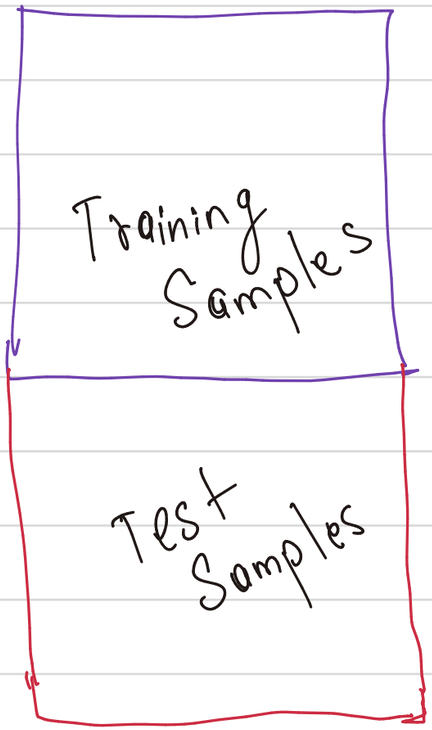
Randomly Split the data that are available into a train and a test set.

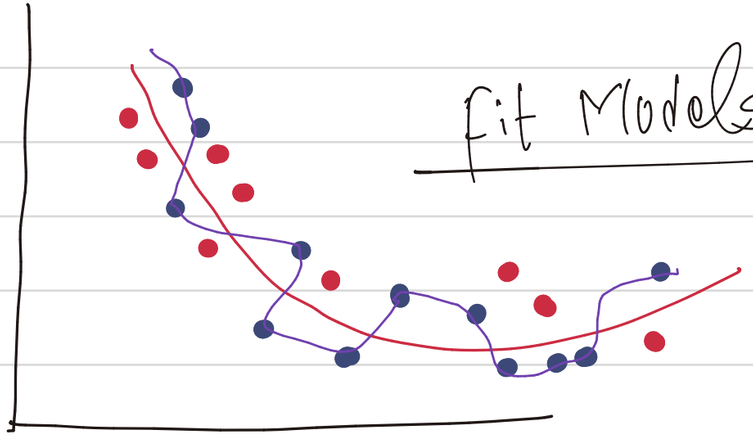
Fit a collection of models, all of different Complexities, on the training set

Look at the performance of those models on the test set

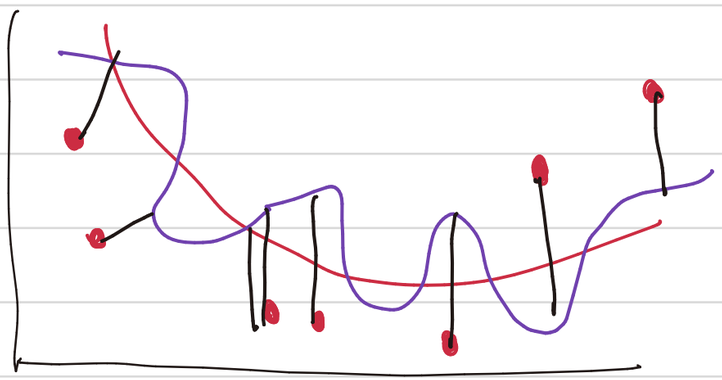
Choose the model with the best performance on the test set.

Split Data.





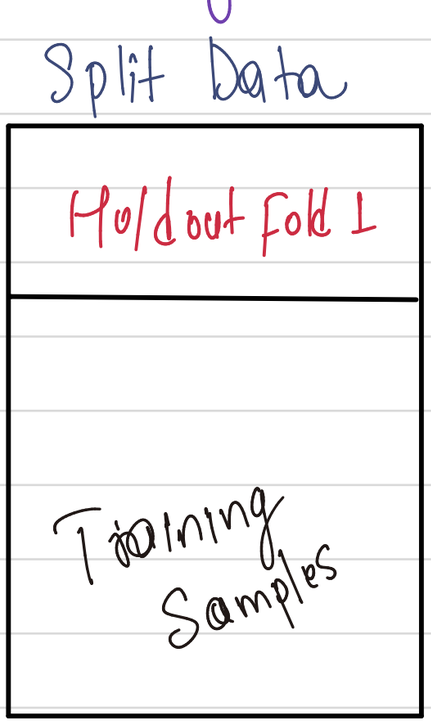
fit Models

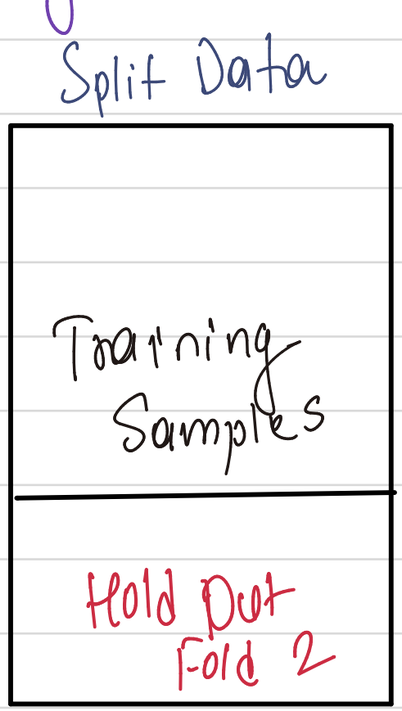


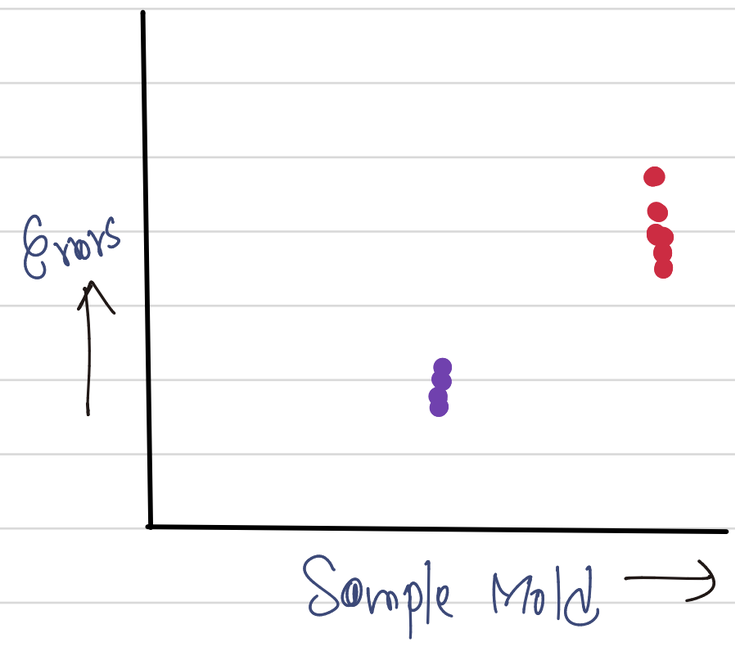
Evaluate Errors.

We only train the training Samples

A more Sophisticated appoarch is to use cross-validation. Instead of using train/test split. We can group the data into K-different "folds". For each Complexity level, we can train K each time leaving out one of the folds. Performance of these models on the estimating the appropriate Complexity level.







How does Cross Validation work?

1) Data Splits into k-folds (K-Subsets)

2) Training and Testing:

Model is trained on K-1 folds (eg: 4 out of 5 folds)

and (tested on the remaining 1 fold)

this process is repeated K times, each time using a different fold as the test set and the remaining K-1 folds for training

3) Average Results:

After all K-iterations, the model's performance (eg: accuracy

, error) is averaged across all folds to give a more reliable estimate of its generalization ability.

Theoretical part:

There are a few subtleties related to train/test splits and cross-validation that are worth knowing -

- We want the held-out data to be as representative of new, real-world data as possible.

Sometimes, randomly split data will not be representative. For example, maybe the data are not collected over time.

A more "honest" split would be into past vs. future samples

starting at a few different timepoints, training on past and evaluating on future. The sample issue occurs if we plan on applying a model to a new geographic context or market segment, for example.

Cross-validation can be impractical on larger datasets. For this reason, we often see only a single train/test split for evaluation of algorithms in large data settings.

We incur a bias when splitting the data. The fact that we split the data means that we aren't using all the available data, which leads to slightly lower performance.

if it hurts more complex models more than simpler ones, than it can even lead to slightly incorrect selection.

After having choosen a given model complexity, it is worth retraining that model on the full available dataset.

Randomly Split the data before training the model:

Example of future and past Splitting of dataset is on predicting the Stock price, Geographic data-set,

Be-careful with the data Splitting like using Subset of datasets, how and what to use and where to use dataset?

After the coding part for both Cross-Validation and train-Split-Test

Note!:

Related to overfitting, there is important phenomenon called the bias-variance trade off.

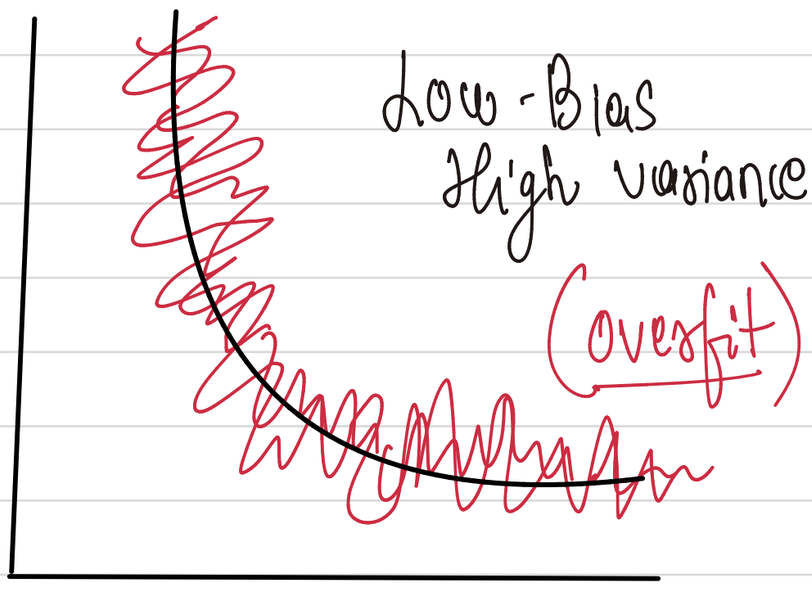
To understand this bias-variance trade off, imagine being able to sample hundreds of datasets similar to the one that is available.

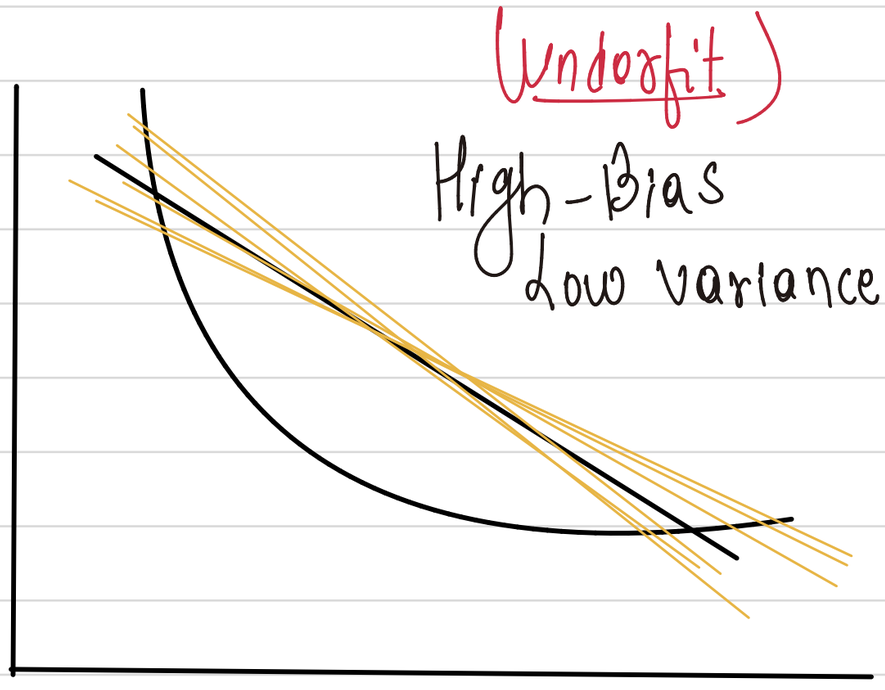
On each of these datasets we can train a new model (at a given complexity level). then, we can define

1. Bias
2. Variance

① Bias: Average the predictions from models trained across each of the imaginery datasets. How far off they are from the truth, on average?

② Variance: How similar are the predictions from across the different runs?





Ideally, we would have low values for both bias and variance, Since they both Contribute to the performance on the test-set.

In practice though there is trade-off.

often, the high-powered models that tend to the closest to the truth on average might be far off on any individual run (high variance). Conversely, overly simple models that are stable from the run to run might be Consistently incorrect in certain regions (bias).

Models with high variance but low bias tend to be overfit &
Models with low variance but high bias tend to be underfit.

Models that have good test or Cross-validation errors have found a good compromise between bias and variance.

Tips: Starts with the Complicated or Complex Models
Starts to trim and optimize this, then we could have more generalized Model. Rather than starting from Simplex to the Complex Model.

Choosing the Models with the right Compleaity.

Logistic Regression Need & loss Function

① There was no discontinuous function in the linear function because of its threshold functions.

② $h_w(x)$ is not differentiable this makes learning with perceptron very Complicated.

③ We can use logistic function (for logit) and can softened the hard threshold functions- approx Hard function with Continuous and differentiable functions.

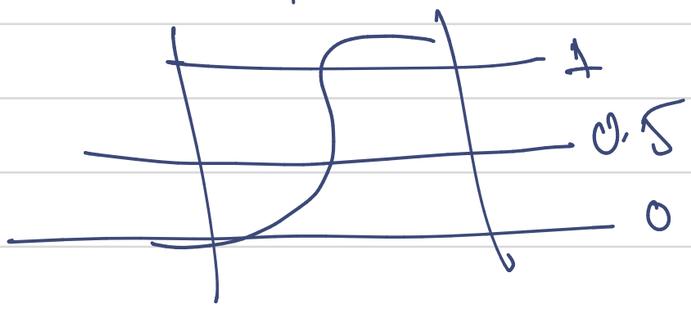
So, the soft threshold functions -

$$\text{Logistic}(z) = \frac{1}{1+e^{-z}}$$

Where $z = wx$ = weight * input

→ With these our output is no longer 0 or 1 (linear fonchon used before as perceptron learning with XOR and AND gates).

→ Output is between 0 to 1 such that output can be 0.23, 0.5, 0.78, 0.9 so



$g \Rightarrow$ Logistic function

$g' \Rightarrow$ derivative of Logistic Regression

So to $h_w(x)$ Continuous we use loss function

Chain Rule $\Rightarrow$ $\frac{\partial g\left(f(x)\right)}{\partial x} \Rightarrow g'\left(f(x)\right) \cdot \frac{\partial f(x)}{\partial x}$

Derivation of loss function

$$\frac{\partial}{\partial w_i} \text{loss}(w) \Rightarrow \frac{\partial}{\partial w_i} \left(y - h_w(x)\right)^2$$

$$\frac{\partial}{\partial w_i} \text{loss}(w) \Rightarrow \frac{\partial}{\partial w_i} \underset{(z)}{(y - h_w(x))^2} \quad \left[\frac{\partial}{\partial z} z^2 \Rightarrow 2z \right]$$

$$\Rightarrow 2(y - h_w(x)) \times \frac{\partial}{\partial w_i} (y - h_w(x))$$

$$\Rightarrow -2(y - h_w(x)) \times g'(w.x) \times \frac{\partial}{\partial w_i} w.x$$

$$\Rightarrow \quad -2(y - h_w(x)) \times g'(w.x) \times x_i$$

derivative of 'g' $\rightarrow$ g'

$$g \Rightarrow \frac{1}{1+e^{-z}}$$

$$g'(z) \Rightarrow \quad g(z)\ (1 - g(z))$$

So, we have,

$$g'(w.x) \Rightarrow \quad g(w.x)\ (1 - g(w.x))$$

$$\Rightarrow h_w(x) \cdot\ (1 - h_w(x))$$

Weight update for minimizing the loss is -

$$w_i \leftarrow w_i + \alpha \left(y - h_w(x)\right) \times h_w(x) \times \left(1 - h_w(x)\right) \cdot x_i$$

So, $w_i \leftarrow w_i + \alpha J$

where $J \Rightarrow (y - h_w(x)) \times h_w(x) \times (1 - h_w(x)) \cdot x_i$

Logistic Regression are better for larger dataset where data are noisy and non-separable

$h_w(x)$ where

x is the input feature vector

$w \Rightarrow$ weight vector (model parameter)

$h_w(x)$ output (predicted probability) from the logistic Regression model of given input (x).

Squared Error loss: $(y - y')^2$

where y' is the predicted outcome & y is the actual outcome

So,

$$\text{loss}(w) = (y - h_w(x))^2$$

y can be either '0' or '1' & $h_w(x)$ is the predicted value $[0,1]$.

We Calculate gradient

$$\frac{\partial}{\partial w_j} \text{loss}(w)$$

Expand the Loss function:

$$\text{loss}(w) = \sum (y - h_w(x))^2$$

So, we apply Chain Rule-

$$\frac{\partial}{\partial 2} 2^2 = 2z$$

So we get,

$$\frac{\partial}{\partial w_i}\left(y - h_w(x)\right)^2 \Rightarrow 2\left(y - h_w(x)\right)\cdot\frac{\partial}{\partial w_i}\left(y - h_w(x)\right)$$

Because taking derivative, we take first after function derivative-

$$x^2 \Rightarrow 2x$$

$$\left(y - h_w(x)\right)^2 \Rightarrow 2\left(y - h_w(x)\right)$$

But we have inner function too $\left(y - h_w(x)\right)$

Outer function $z^2 \Rightarrow 2z$

$$2 = (y - h_w(x)) \Rightarrow 2(y - h_w(x))$$

Inner function:

$$z \Rightarrow y - h_w(x)$$

derivative with w_j -

$$\frac{\partial}{\partial w_j}(y - h_w(x)) \Rightarrow \frac{-h_w(x)\partial}{\partial w_j}$$

So, $\frac{\partial}{\partial w_i}(y - h_w(x))^2 \Rightarrow 2(y - h_w(x)) \cdot \frac{\partial}{\partial w_i}(y - h_w(x))$

$\Rightarrow 2(y - h_w(x)) \cdot -\frac{\partial h_w}{\partial w_i}$

$\Rightarrow -2(y - h_w(x)) \cdot \frac{\partial h_w}{\partial w_i}$

$\Rightarrow -2(y - h_w(x)) \cdot g'(wx) \cdot x_i$

and $h_w(x) \Rightarrow g(w.x)$.

[$\frac{\partial}{\partial w_i}(y - h_w(x))$ is $-\frac{\partial h_w}{\partial w_i}$]

[$\frac{\partial h_w}{\partial w_i}$ is $g'(w,x)$]