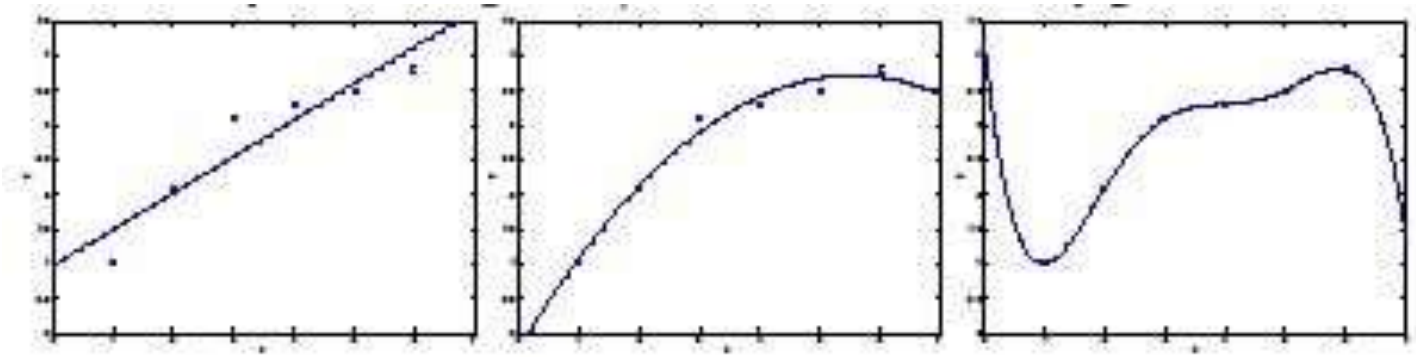
MACHINE LEARNING THEORY

Predicting of y from x $\in \Re$

$y = w_0 + w_1 x$

$y = w_0 + w_1 x + w_2 x^2$

$y = \sum_{j=0}^{5} w_j x^j$



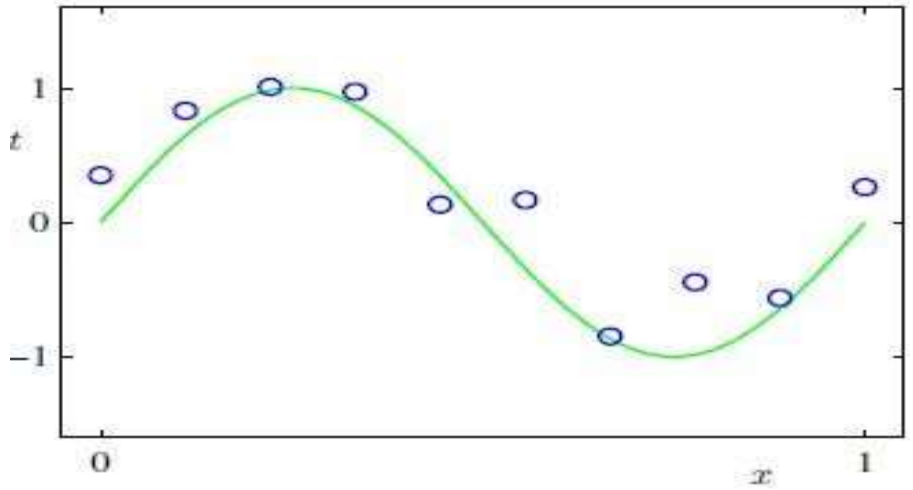
Data points do not lie on the straight line

Added an extra feature x^2. Slightly better fit the data

More features make better fitting

But higher order polynomial or more features used for modeling not a good predictor.

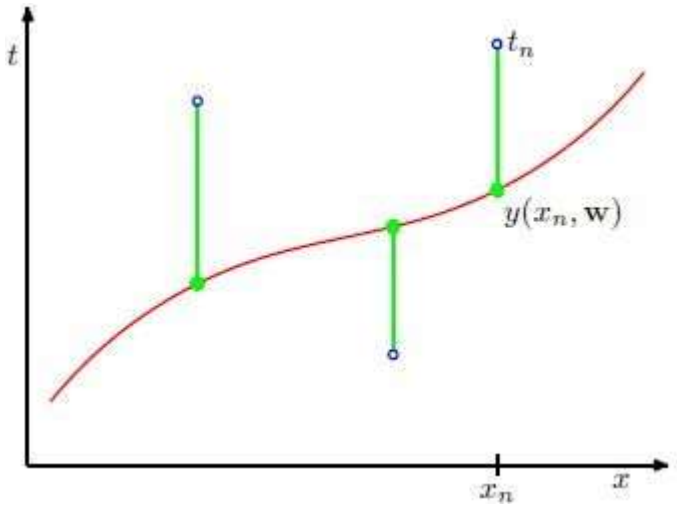
- Selection of features are important for good prediction



Modeling of function sin(2πx)

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1} \{y(x_n, \mathbf{w}) - t_n\}^2$$



Error function is a nonnegative quantity that would be zero if, and only if, the function y(*x*,**w**) *passes exactly through each data point.*

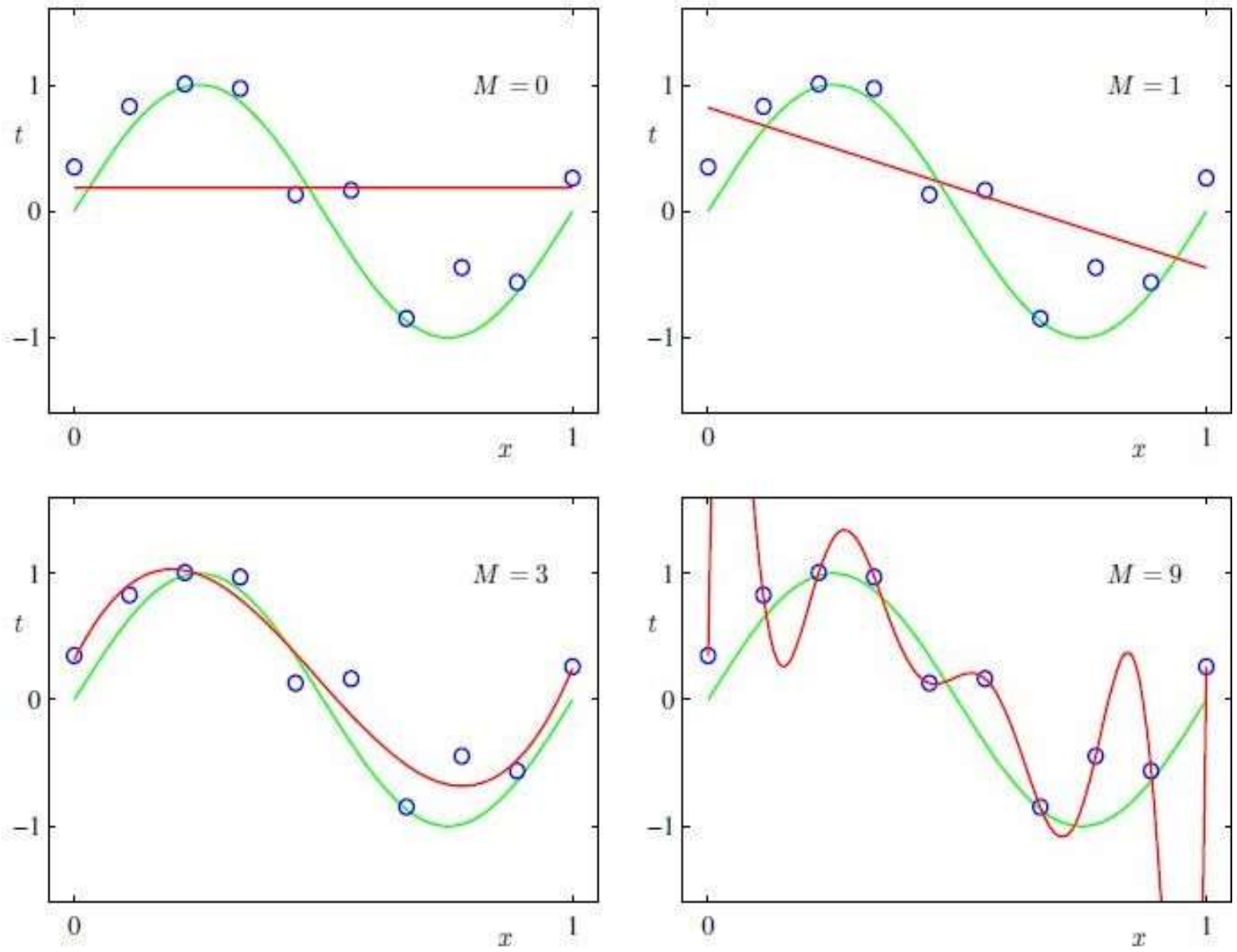
Error Function

- We can solve the curve fitting problem by choosing the value of **w** for which ***E*(*w*) is as small as possible**.

- Because the error function is a quadratic function of the coefficients **w,** the derivatives with respect to the coefficients will be linear in the elements of **w.**

- So the minimization of the error function has a unique solution, denoted by **w***.

The problem of choosing the order *M* of the polynomial, Important concept i.e. model selection.



Model Selection

- The constant ($M = 0$) and first order ($M = 1$) polynomials poor fit the training data and consequently poor representations of the function $\sin(2\pi x)$.
- The third order ($M = 3$) polynomial best fit to the function $\sin(2\pi x)$ but not all the training data.
- Higher order polynomial ($M = 9$), obtain an excellent fit to the training data, and $E(\boldsymbol{w}) = 0$.
- However, the fitted curve oscillates wildly and gives a very poor representation of the function $\sin(2\pi x)$.
- This behaviour is known as **over-fitting**.

- The goal is to achieve good generalization by making accurate predictions for new data.
- Generalization performance on M is measured by the test data.
- For each choice of M, we evaluate $E(\boldsymbol{w}^*)$ for the training data, and also evaluate $E(\boldsymbol{w}^*)$ for the test data set.
- It is sometimes more convenient to use the root-mean-square in which the division by N allows us to compare different sizes of data sets on an equal scale.

Root Mean Square Error $=$ $\sqrt{2E(w^*)/N}$

coefficients **w***

	$M=0$	$M=1$	$M=6$	$M=9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43

Coefficients **w***

- As M increases, the magnitude of the coefficients become larger.
- For $M = 9$ polynomial, the coefficients are finely tuned to the data with large positive and negative values so that the corresponding polynomial function matches each of the data points exactly.
- Between the data points (particularly near the ends of the range) the function exhibits the large oscillations.

Generalization and Over-fitting

- Best model is one doing accurate prediction for all data.
- Such model can **generalise** other than the training instances.
- Selection of Model is challenging.
- More complex model makes closer training data points up to a certain limit, beyond which the prediction accuracy fall.
- If the model does not have the generalization capacity the problem is called **overfitting**.

Why OVERFITTING

- One of the major aspects of training the model is avoiding *overfitting.*
- **Overfitting** happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data due to detailing of large training data.
- The model or Machine learning algorithm will have a low accuracy if it is *overfitted.*
- Overfitting occurs due to complexity which have more freedom in building the model based on the dataset and therefore they can really build unrealistic models.

Bias

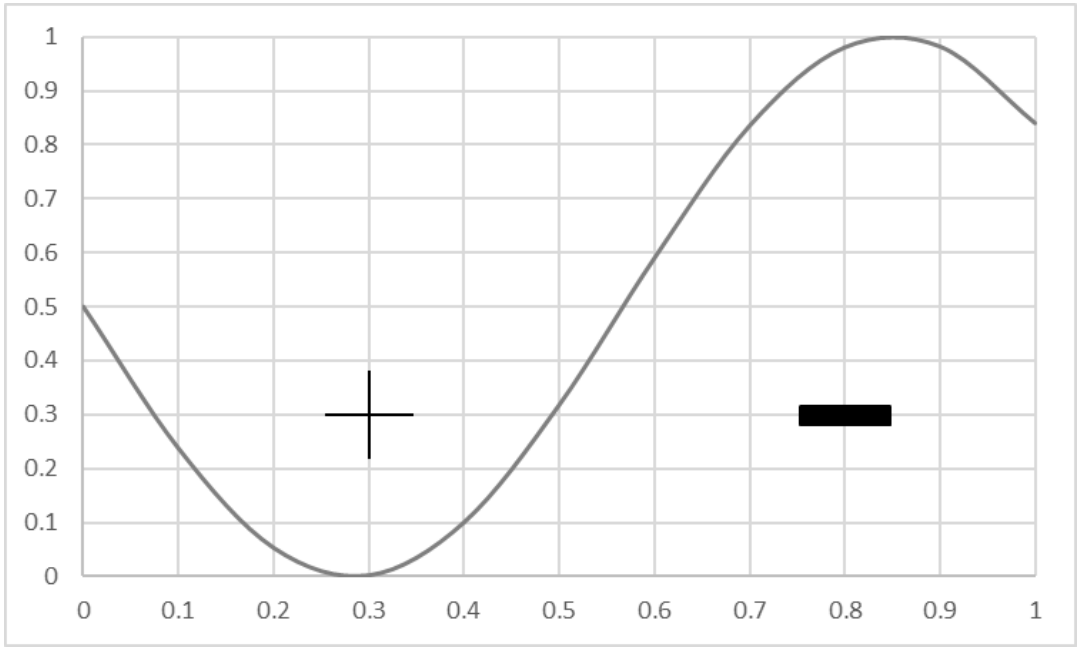
- Bias refers to how correct the model is.
- A very simple model that makes a lot of mistakes is said to have high bias.
- A very complicated model that does well on its training data is said to have low bias.
- The idea of having bias about the model giving importance to some of the features in order to generalize better for the larger dataset with various other features.
- **Bias** in machine learning is a type of error in which certain features of a **dataset** are more heavily weighted.

Variance

- Variance which describes how much a prediction could potentially vary if one of the predictors changes slightly.
- Simplicity of the model makes its predictions change slowly with predictor value, so it has low variance, high bias.
- On the other hand, complicated, low bias model likely fits the training data very well and so predictions vary wildly even predictor values change slightly.
- This means this model has high variance, and it will not generalize to new/unseen data well.

Visualizing Bias

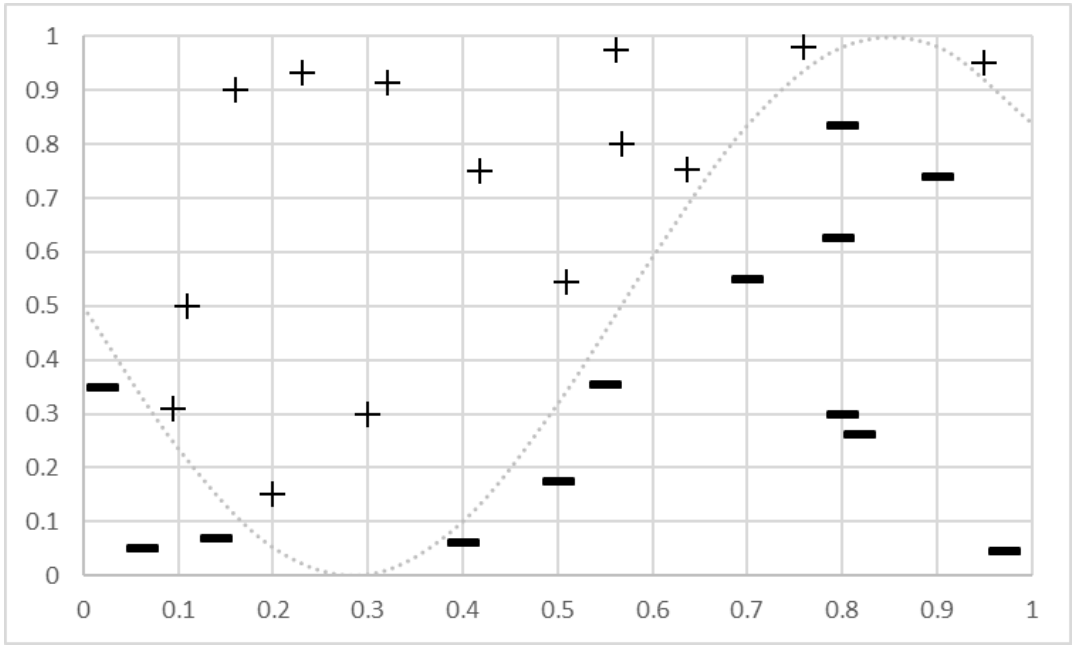
Goal: produce a model that matches this concept



True Concept

Visualizing Bias

Goal: produce a model that matches this concept
Training Data for the concept

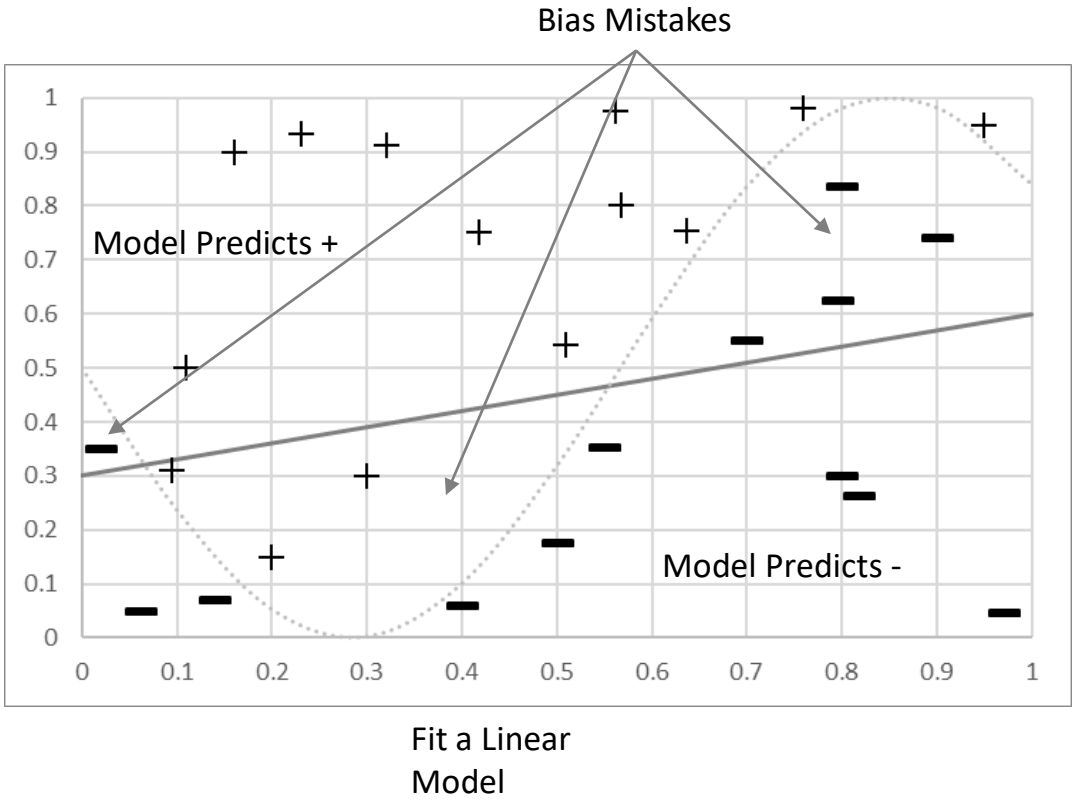


Training Data

Visualizing Bias

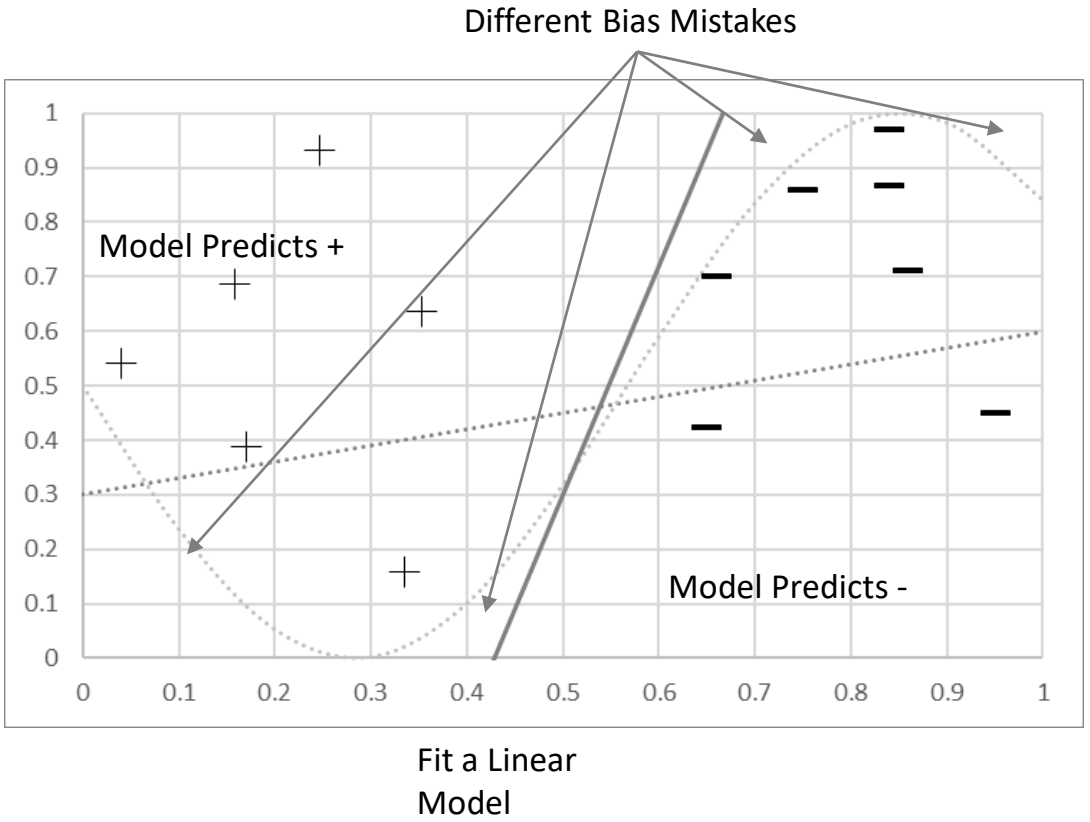
Goal: produce a model that matches this concept
Training Data for concept

Bias: Can't represent it…



Visualizing Variance

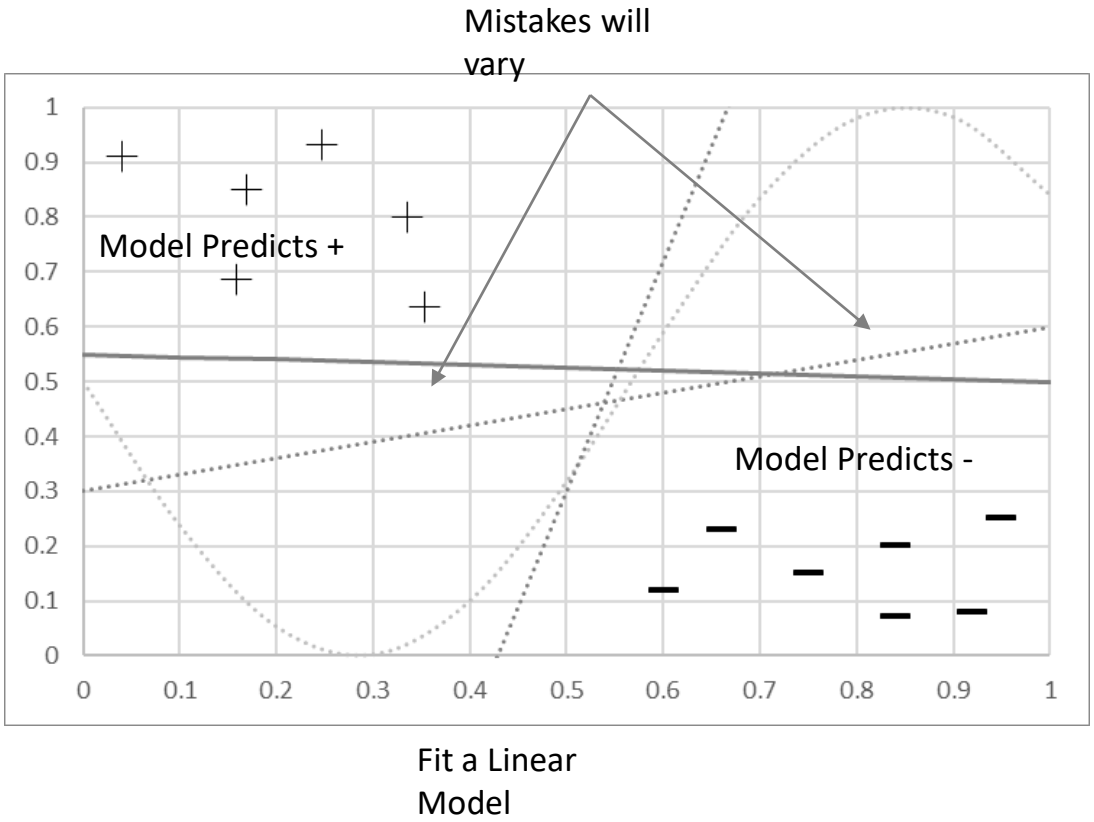
Goal: produce a model that matches this concept
New data, new model



Visualizing Variance

Goal: produce a model that matches this concept
New data, new model
New data, new model…

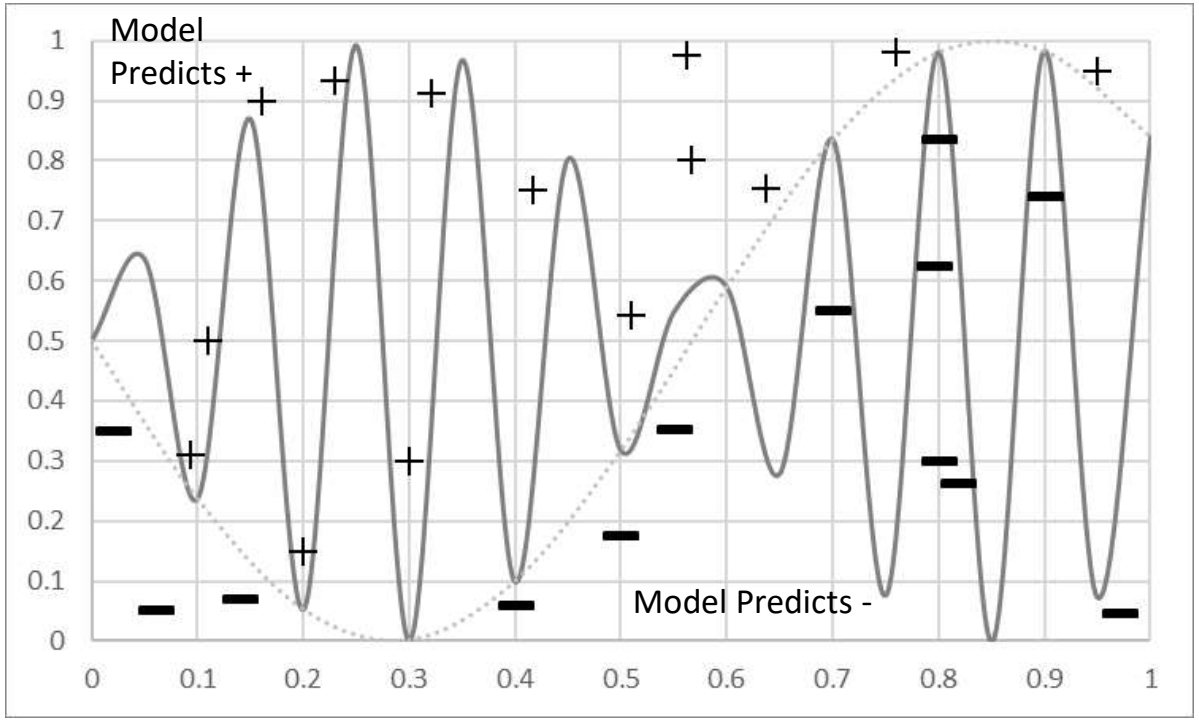
Variance: Sensitivity to changes & noise



Bias and Variance: More Powerful Model

Powerful Models can represent complex concepts

No Mistakes!

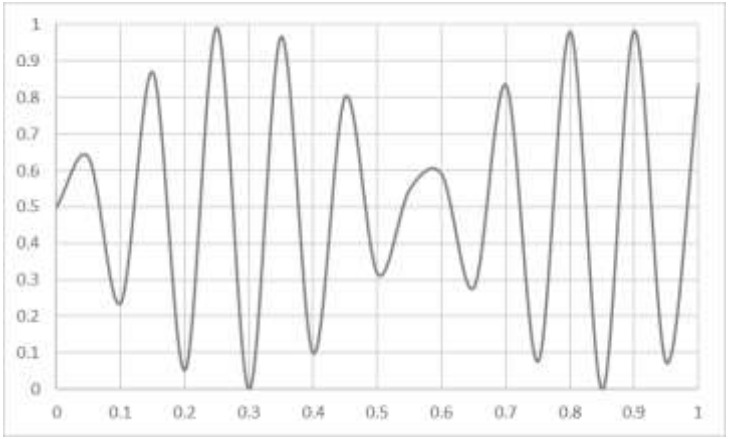


Overfitting vs Underfitting

Overfitting

Fitting the data too well

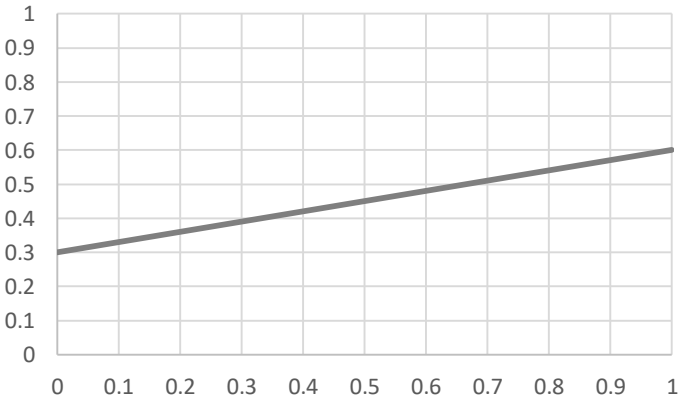
- Features are noisy / uncorrelated to concept
- Modeling process very sensitive (powerful)
- Too much search



Underfitting

Learning too little of the true concept

- Features don't capture concept
- Too much bias in model
- Too little search to fit model



Overfitting

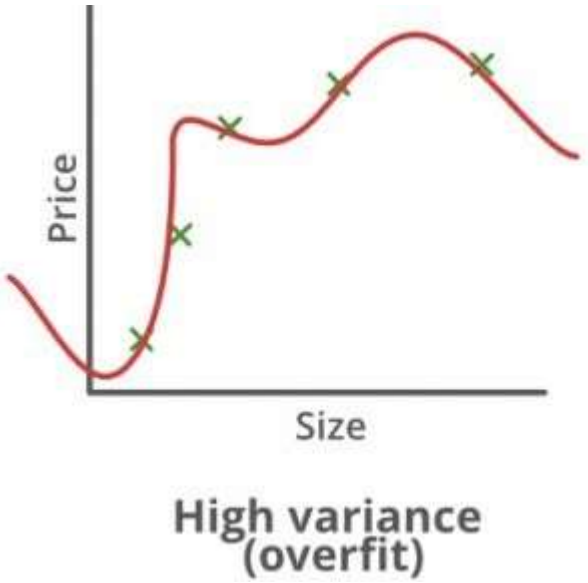
O•verfitting- Low Bias, High Variance

Avoid overfitting

1. Increase training data.

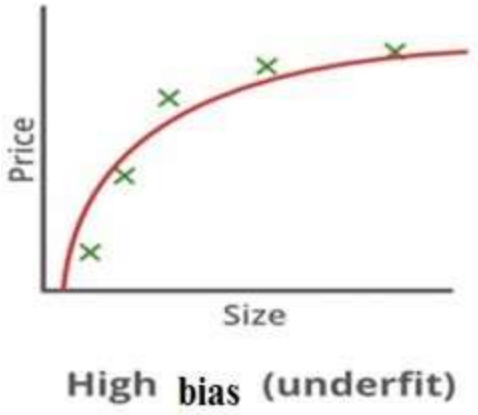
2. Reduce model complexity.

3. Regularization



Underfitting

- A model is said to have underfitting when it cannot capture the underlying trend of the training data and generalization ability.
- Causes due to less data but more features to build the model and we try to build a linear model with a non-linear data.
- Underfitting can be avoided by using more data and reducing the features by feature selection.



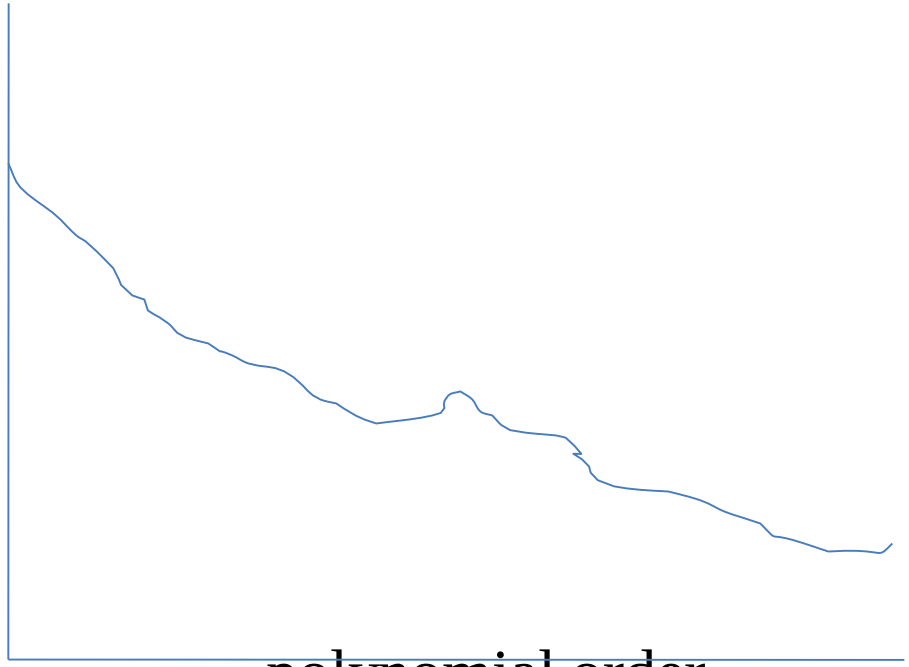
Underfitting – High bias and low variance

Validation

- Use a second dataset, often referred as a validation set.
- Train the model with training dataset and then compute loss using validation data set.
- The training loss decreases monotonically with the order of polynomial i.e. model complexity
- Validation loss increases with the complexity of the model, suggesting that a first order model has best generalization ability.

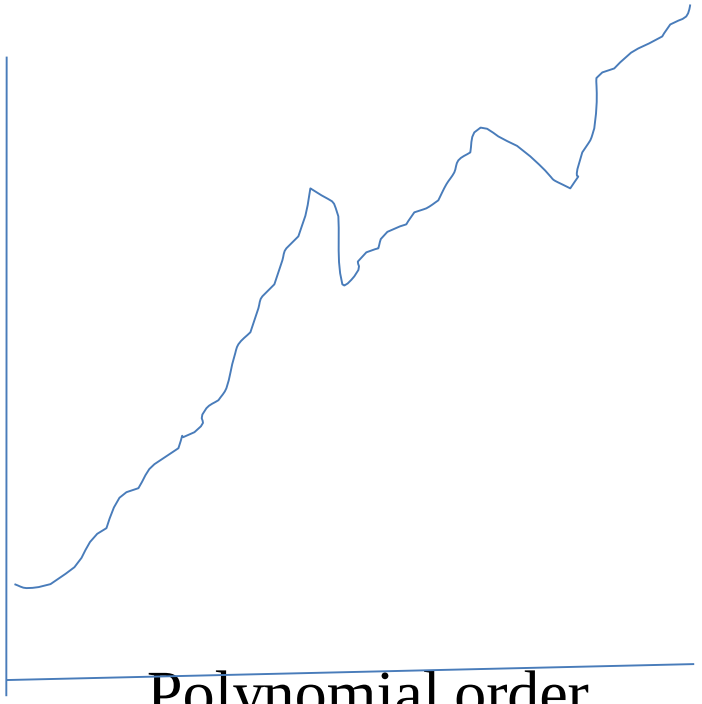
Loss with polynomial order

Training Loss



polynomial order

Validation Loss



Polynomial order

Cross Validation

- One of the ways of avoiding overfitting is using cross validation, that helps in estimating the error over test set.
- K-fold cross validation split the data into equal K blocks or subsets.
- Each time, one of the K subsets is used as the test set/ validation set and the other K-1 subsets are used as training set.
- The error estimation is averaged over all K trials.
- Averaging over K loss value gives final loss

Cross Validation

- Every data point gets to be in a validation set exactly once, and gets to be in a training set *K*-1 times.
- This significantly reduces bias as we are using most of the data for fitting, and also significantly reduces variance as most of the data is also being used in validation set.
- Interchanging the training and test sets also adds to the effectiveness of this method.
- As a general rule and empirical evidence, K = 5 or 10 is generally preferred.

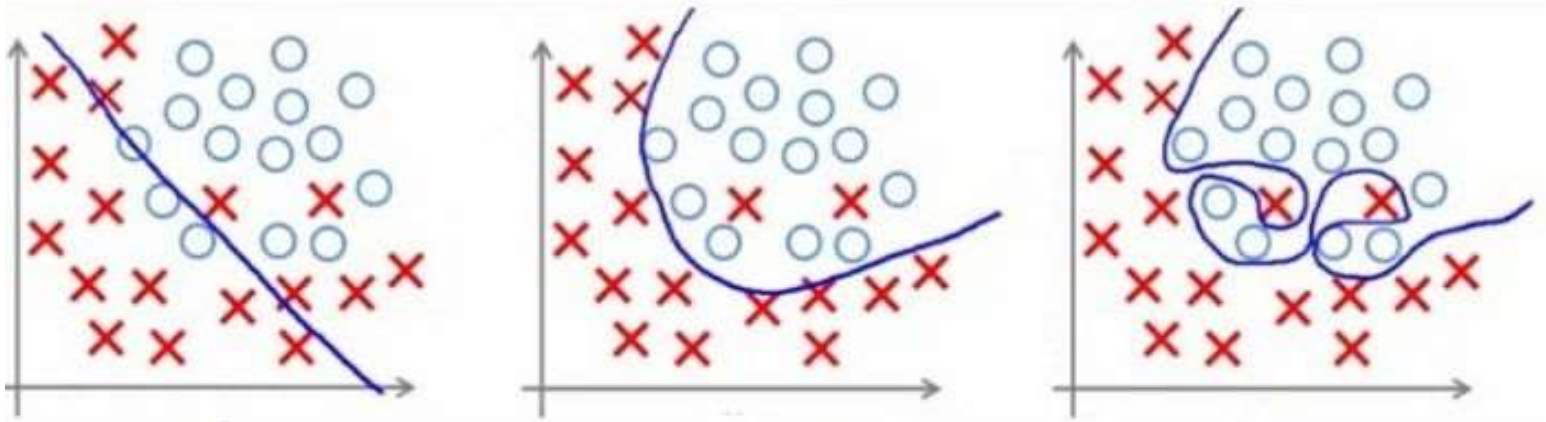
Leave one out

- Extreme case when K=N (no. of observations)
- Each data is held out in turn and used to test a model trained on the other N-1 objects or observations.
- Average squared validation loss: $L^{CV} = 1/N \sum_n (t_n - \mathbf{w}^T_n \mathbf{x}_n)^2$

Where $\mathbf{w}_n$ is the estimate of the parameters without the n-th training example.

- In order to avoid computational complexity K<<N, generally 10 fold, i.e. 10% for validation and 90% for training

Fitting



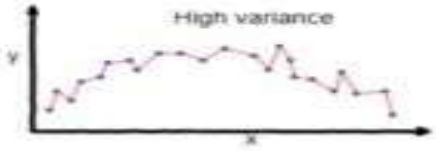
Under-fitting

(too simple to explain the variance)

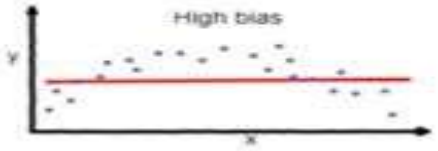
Appropriate-fitting

Over-fitting

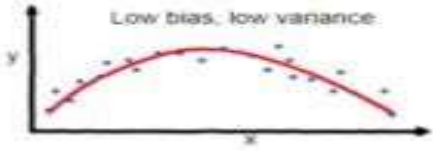
(forcefitting -- too good to be true)



overfitting



underfitting

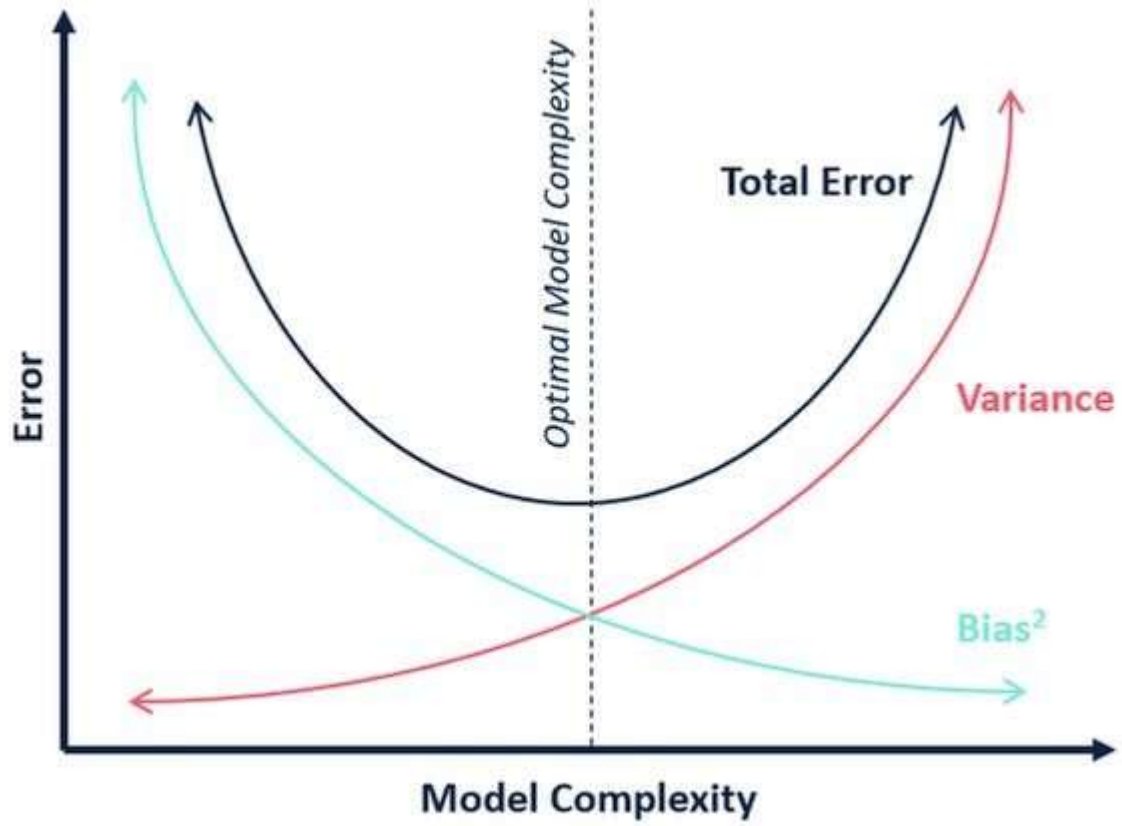


Good balance

Bias Vs Variance

- The low-bias/high-variance model exhibits overfitting, in which the model has too many terms and explains random noise in the data on top of the overall trend.
- The high-bias/low-variance model exhibits underfitting, in which the model is too simple/has too few terms to properly describe the trend seen in the data. Again, the model will struggle on new data.
- We have the proper number of terms to describe the trend without fitting to the noise.
- We therefore need some sort of feature selection in which predictors with no relationship with the dependent variable are not influential in the final model.

The bias-variance tradeoff

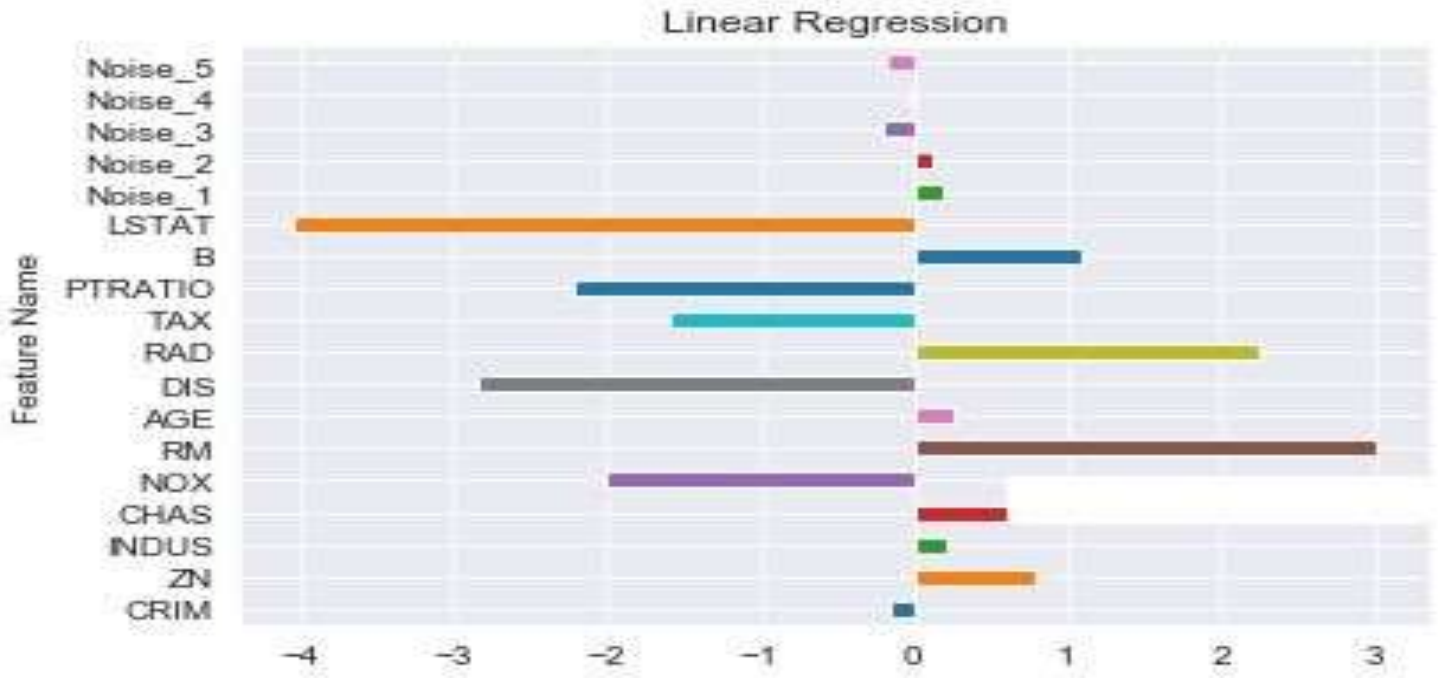


The total error of the model is composed of three terms: the $(\text{bias})^2$, the variance, and an irreducible error term.

Bias vs. Variance

- Weights for some predictors have little predictive power, results high-variance, low bias model.
- We improve the model by trading variance with bias to reduce overall error.
- This trade comes in the form of **regularization.**
- We modify the cost function to restrict the values of weights, which allows to trade excessive variance, potentially reducing overall error.

Linear Regression



The model assigns non-zero values to the noise features, despite none of them having any predictive power. Noise features have values similar to some of the real features in the dataset.

Example

- Hyundai: This column indicates the money spent on advertising the Hyundai cars in the given market.
- Maruti – Similarly, money spent on advertising by Maruti car.

	Hyundai(Thousands of dollars)	Maruti (Thousands of dollars)	Mahindra (Thousands of dollars)	Sales(Thousa of dollars)
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

- Mahindra – Similarly, money spent on advertising by Mahindra car.
- Sales – This column indicates the sales of cars in the given market. (Value of the sales in thousands)

Example

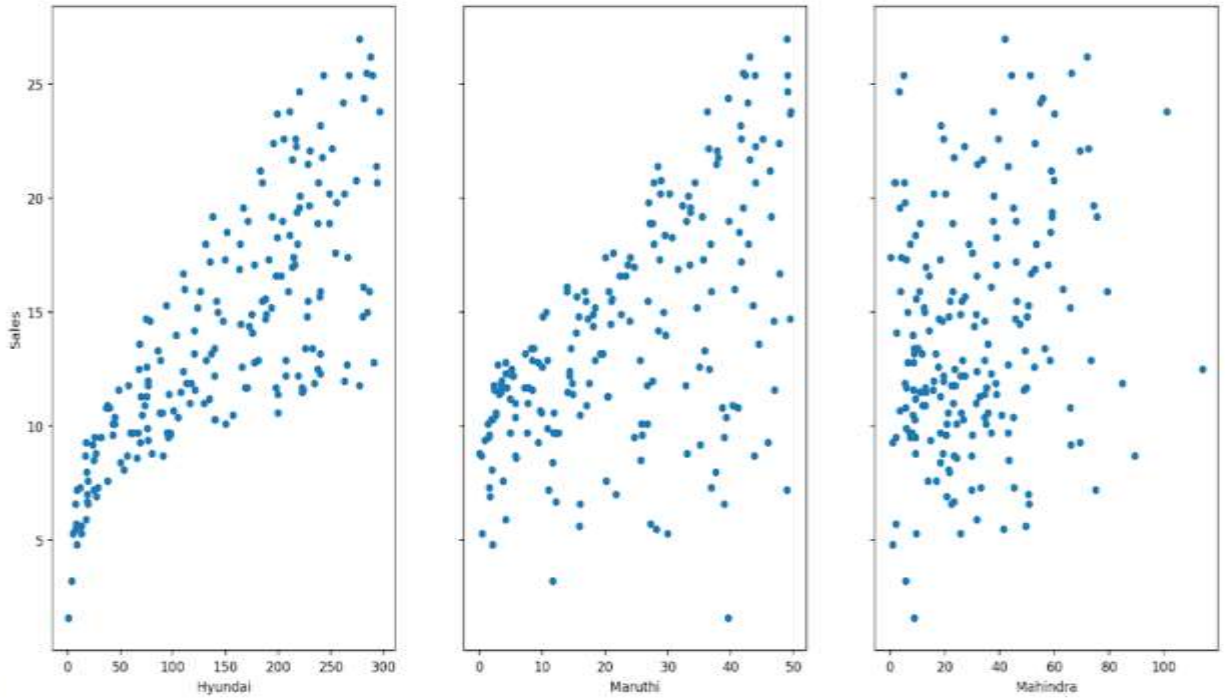
- Let's visualize the relationship between the features and the sales response using scatterplots.

```
In [6]: fig, axs = plt.subplots(1, 3, sharey=True)
        data.plot(kind='scatter', x='Hyundai', y='Sales', ax=axs[0], figsize=(16, 8))
        data.plot(kind='scatter', x='Maruthi', y='Sales', ax=axs[1])
        data.plot(kind='scatter', x='Mahindra', y='Sales', ax=axs[2])
```

1.886 seconds

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1661ab6a90>
```

Out[6]:



- Now by looking into the above scatterplots we can easily say that Hyundai advertising and sales have a strong relationship.
- However, Maruti's and Mahindra's datapoints look scattered all over the graph that implies that they have a weak relationship between advertisement and sales.

Example

Now let's Estimate the model coefficients for Linear Regression by using single feature to predict quantitative response. It takes the following form:

$$\mathbf{y = a + bx}$$

Where, y will be the response

X will be the feature

a is the intercept

b is the coefficient for x, a & b are called Model coefficient.

To calculate coefficients, we will use the least square criterion, which means we will find a line that will decrease the sum of squared errors.

Example

- From previous step we got the value of A and B. we will use the model to predict the future sales of Hyundai cars.
- Let's say in the new market Hyundai is spending 50 thousand dollars in advertising. That means the new value of X will be 50.
- Now using Y = A + BX to predict the new value.

In [11]:
```
# manually calculate the prediction y = a + bx
7.032594 + 0.047537*50
```
0.006 seconds 8

Out[11]: 9.409444

In [12]:
```
X_new = pd.DataFrame({"Hyundai": [50]})
X_new.head()
```
0.058 seconds 9

Out[12]:

	Hyundai
0	50

In [13]:
```
lm.predict(X_new)
```
0.005 seconds 10

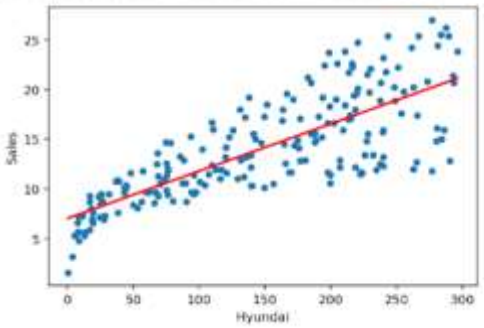
Out[13]:
```
0    9.409426
dtype: float64
```

Example

- Let’s plot the observed data graph and the least square line using preds value and new x value.

```
In [16]: data.plot(kind='scatter', x='Hyundai', y='Sales')
         plt.plot(X_new, preds, c='red', linewidth=2)
```

```
[<matplotlib.lines.Line2D at 0x7f16618c7cc0>]
```



Out[16]:

Limitations of Regression Analysis :

Parameter instability : This happens in situations where corelations change over a period of time. This is very common in financial markets where economic,tax,regulatory, and political factors change frequently.

Public knowledge of a specific regression relation may cause a large number of people to react in a similar fashion towards the variables, negating its future usefulness.

If any of the regression assumptions are violated, predicted dependent variables and hypothesis tests will not hold valid.

Regularization

- One way to reduce overfitting is to reduce the number of parameters or features by feature selection approach.
- Instead of reducing the number of parameters, we use all the parameters, but we penalize the non-required, unimportant parameters which do not affect the output.
- *Regularization has no role in optimizing the cost function, it only helps in reducing the overfitting*
- It is used to overcome the low bias and high variance behavior of the model.

Regularization

- Generally, a good model does not give more weight to a particular feature.
- The weights should be evenly distributed, can be achieved by regularization.
- There are two types of regularization as follows:
- L1 Regularization or Lasso Regularization
- L2 Regularization or Ridge Regularization

L1 Regularization or Lasso Regularization

- L1 Regularization or Lasso Regularization adds a penalty to the Loss function.
- The penalty is the sum of the **absolute** values of weights.

$$\text{Loss} = Min(\sum_{i=1}^{n}({}^{t}{}_{i} - w_i x_i)^2 + \lambda \sum_{i=1}^{n} |w_i|)$$

λ is the regularization parameter which decides how much we want to penalize the model.

$$W_{t+1} = W_t - \frac{d(loss)}{dW}$$

L2 Regularization or Ridge Regularization

- L2 Regularization or Ridge Regularization adds a penalty to the Loss function.
- The penalty here is the sum of the **squared** values of weights.

$$Min(\sum_{i=1}^{n}({}^{t}{}_{i} - w_i x_i)^2 + \lambda \sum_{i=1}^{n}(w_i)^2)$$

weights which would non-zero or slightly near to zero would tend to become zero, thus, ineffective.

Regularization

- Regularization parameter (λ) tunes the weights to decide how much we penalize flexibility of the model.

$$L(w) = \frac{1}{2N}\left[\sum_{i=1}^{N}(h_w(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} w_j^2\right]$$

where λ is the regularization parameter

N is the number of samples and *n* is number of features.

- λ Controls the fitting weights. As the magnitudes of the weights increase, there will be an increasing penalty on the cost/Loss function. This penalty is dependent on the squares of the Weights as well as the magnitude of λ.

- Regularization, significantly reduces the variance of the model, without substantial increase in its bias.

Regularization

- As the complexity increases Regularization penalizes the flexibility of our model and finding the optimum parameters.
- Selecting a good value of λ (not 0 or infinity) is critical to our need for finding the correct coefficients.
- It is useful to automatically penalize features that make the model too complex.
- This will decrease the Importance to higher terms and will bring the model towards less complexity.

Linear System

- A linear system of *N* equations with *n* unknowns, writing in linear form:
  $X_{N \times n} W_{n \times 1} = Y_{N \times 1}$ Where X is the data matrix.
- The rows in the X matrix are written as feature vectors of dimension n for each sample and stacked and the individual features are the columns.
- Say, N = *n* so X is square matrix and assuming samples in X are **linearly independent** and we write $W = X^{-1} Y$
- A set of vectors is *linearly independent* if no vector in the set is a linear combination of the other vectors.
- In order for the matrix X to have an inverse, we need to ensure that XW = Y has at most one solution for each value of Y**.**
- W which fits all the training data are not likely to generalize to test data.

Regularization

- Impose regularization to avoid overfitting, or to subsample the data or perform some cross-validation (leave out part of the training data when solving for the parameter W.
- What happens if $N < n$?
- In this case, we have fewer samples than feature dimensions – so less equations than unknowns and we can have infinite number of solutions.
- Increase samples or apply feature selection method to select the features that carry the most weights of the problem.

N > n

- Now we have more data than features, so it is less likely of overfit the training set.
- We will not be able to find an exact solution: a *W* that satisfies linear system equation
- Instead we will search for a W that is best in the least-squares sense.
- W can be obtained by solving the modified system of equations: $XW = Y$

  $X^T X W = X^T Y$; $X^T X$ is a squared matrix of dimension $n \times n$

  $W = (X^T X)^{-1} X^T Y = X^+ Y$ where $(X^T X)^{-1} X^T = X^+$ obtained by optimization of least square Loss function.

Regression with Regularization

- keeping *W* small helps reduce generalization error. We can enforce the constraint to keep *W* small by adding a regularization term to the original loss function.

$$L(w) = \frac{1}{2N}\left[\sum_{i=1}^{N}(h_w(x^{(i)}) - y^{(i)})^2 + \lambda \|W\|^2\right]$$

where λ is the regularization parameter

$$\frac{\partial}{\partial w_j} L(W) = \frac{1}{N}\sum_{i=1}^{N}[(x^{(i)})^T W - y^{(i)}]\left(-x_j^{(i)}\right) + \lambda w_j$$

Matrix Representation

Setting the partial derivative to zero

$$\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} (x^{(i)})^T W + \lambda w_i = \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} y^{(i)}$$

$$\frac{1}{N} X^T X W + \lambda W = \frac{1}{N} X^T Y$$

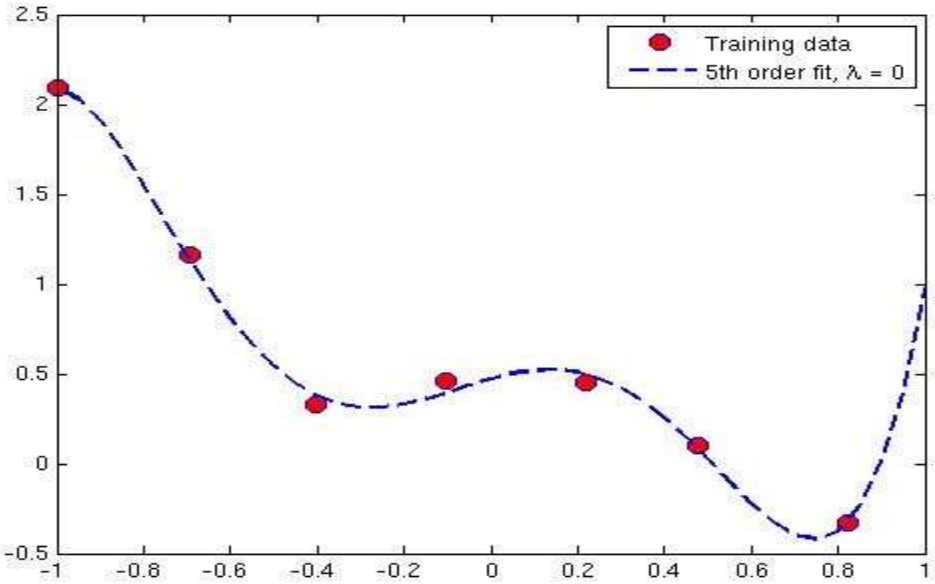
$$\left[\frac{1}{N} X^T X + \lambda I\right] W = \frac{1}{N} X^T Y$$

Solution of W for least square regression with regularization

$$W = [\frac{1}{N} X^T X + \lambda I]^{-1} \frac{1}{n} X^T Y$$

Regularized Least squares

- Complexity of a model is defined as $\sum_i w_i^2$ or $\mathbf{w}^T\mathbf{w}$,
- Rather than minimizing the average squared loss L , we minimize a regularized loss L' : $L' = L + \lambda\mathbf{w}^T\mathbf{w}$
- The second term is the penalized term over complexity of the model.
- λ controls the trade-off between penalizing not fitting the data well (L) and penalizing over complexity of the model ($\mathbf{w}^T\mathbf{w}$)
- when λ is small we prefer to minimize the original cost function, but when λ is large we prefer small weights.

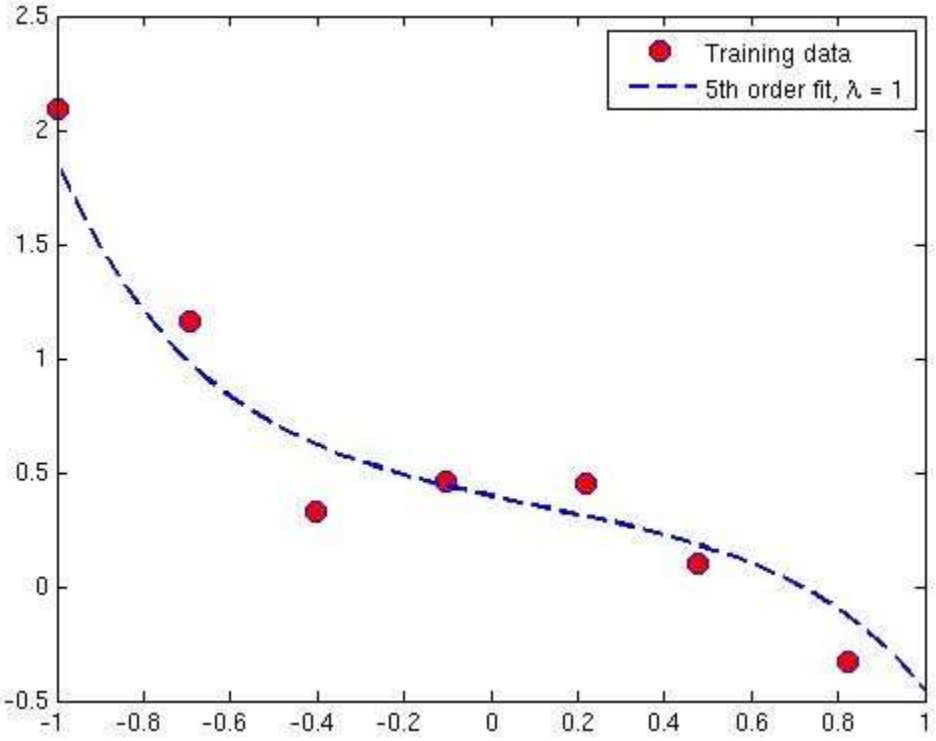


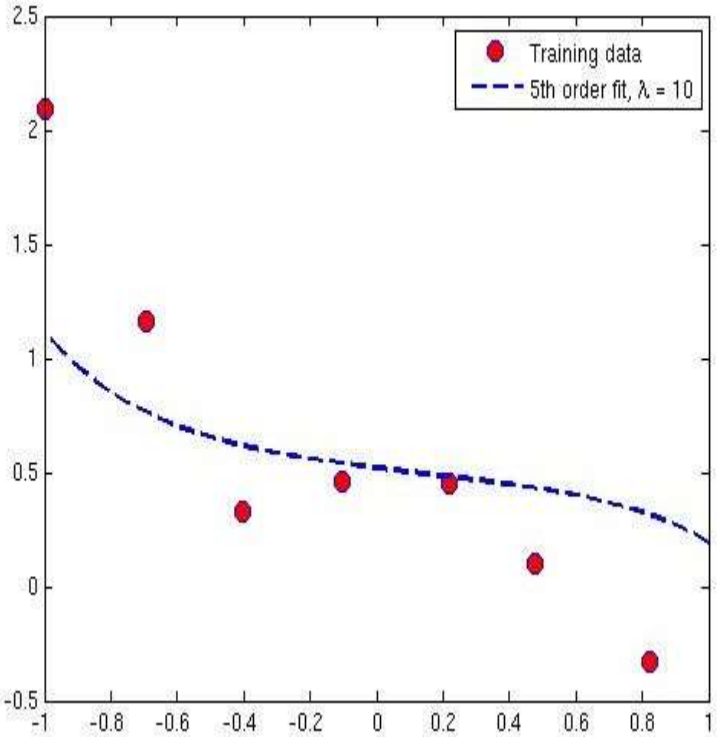
Fitting a straight line might be too simple of an approximation

- Fitting a 5th-order polynomial to a data set of only 7 points, over-fitting is likely to occur.

θ

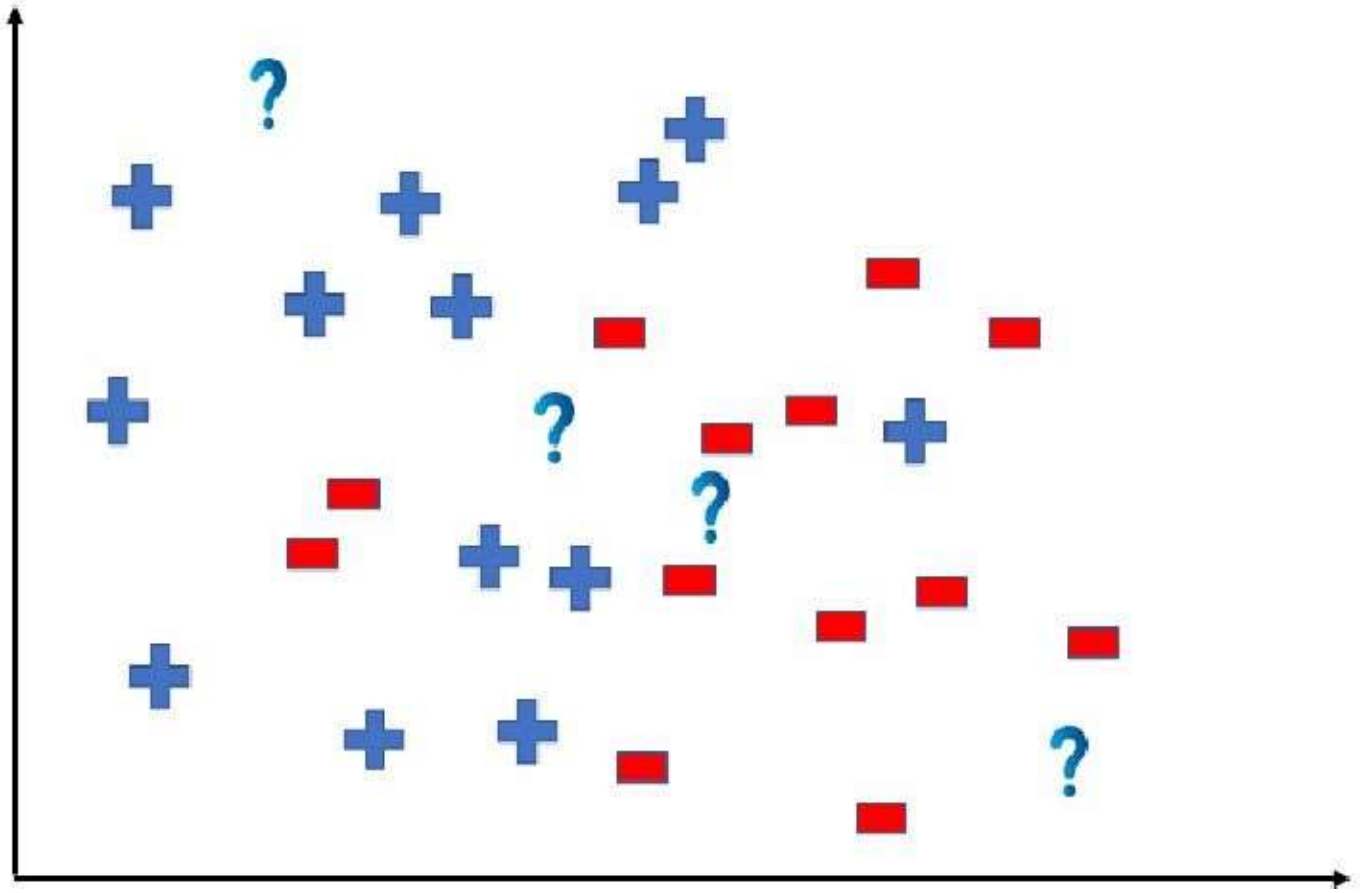
Loss vs regularized parameters





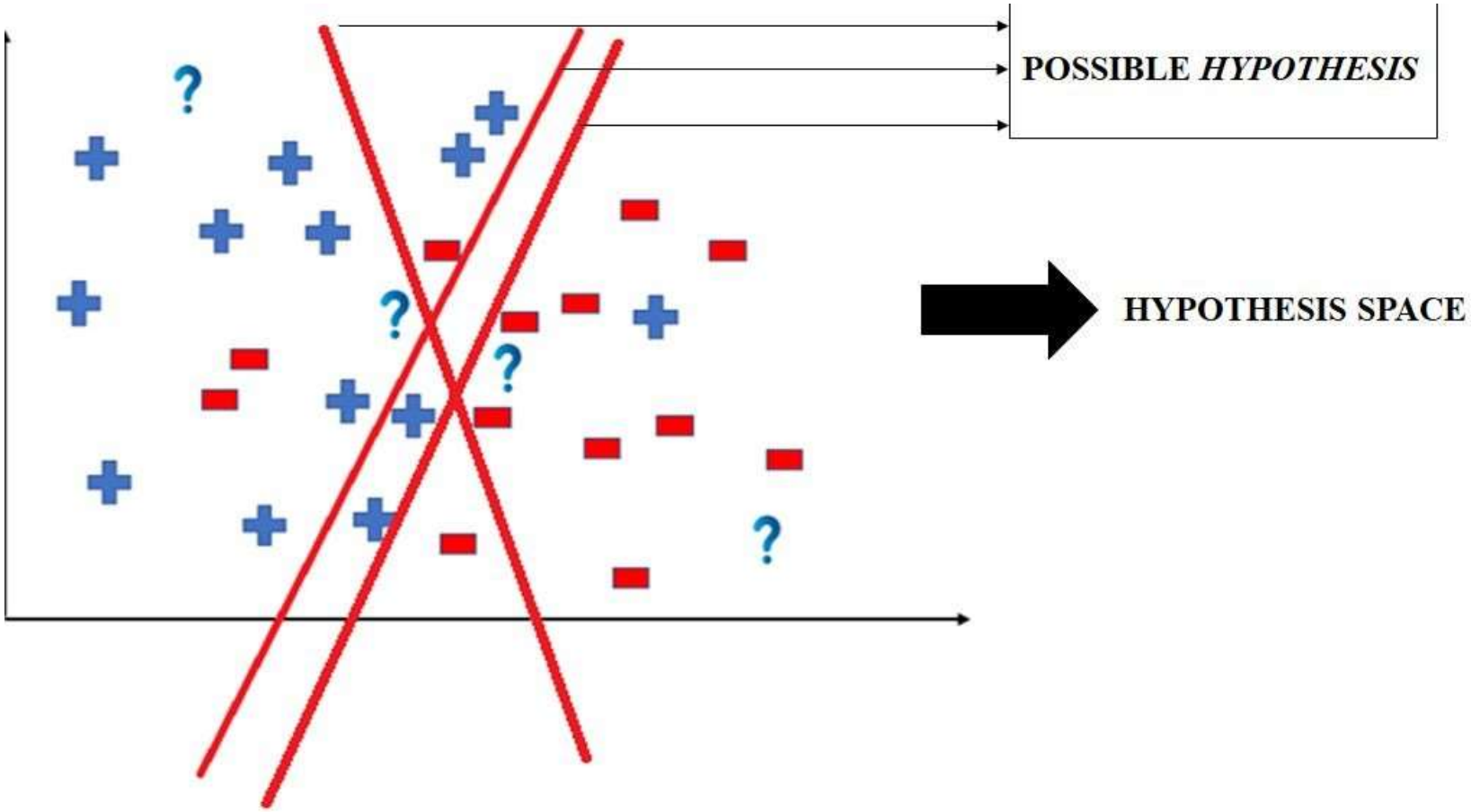
In regularization problems, the goal is to minimize the Loss function with respect to :w

Hypothesis (h)



A hypothesis is a function that best describes the target in supervised machine learning.

Hypothesis Space



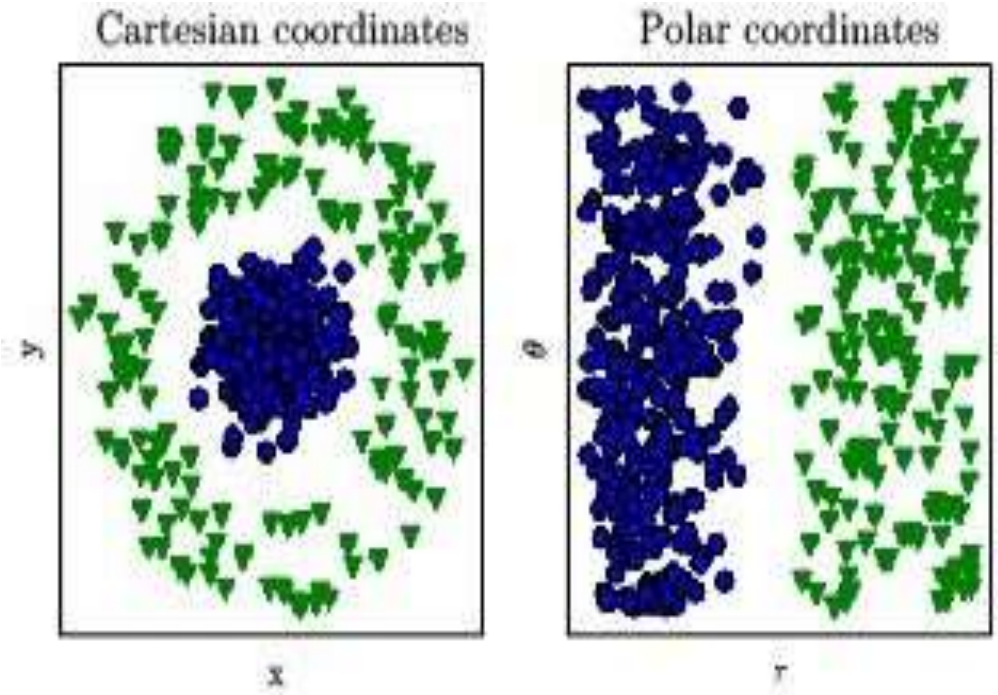
Definition: Hypothesis Space

- In a ML problem, say the input is denoted by x and the output is y
- There should exist a relationship (pattern) between the input and output values. Lets say that y = *f*(x), known as the **target function**.
- Machine learning algorithms try to guess a ``hypothesis'' function *h*(x) that approximates unknown *f*(.).
- The set of all possible hypotheses is known as the Hypothesis set H(.)
- The goal is the learning process is to find the final hypothesis that best approximates the unknown target function *f*(.).

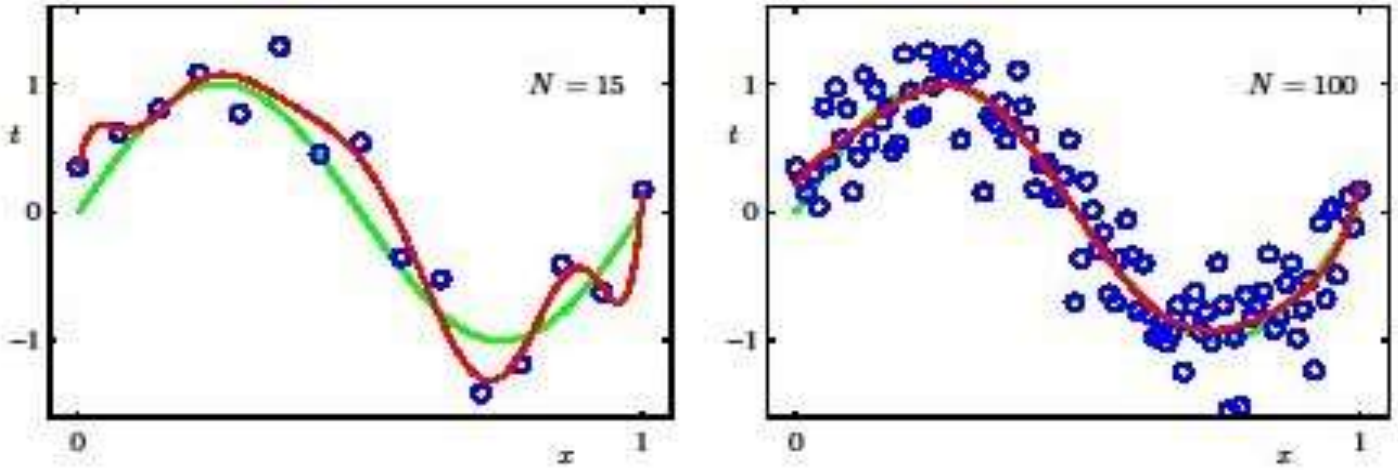
Challenges

- **Representation**: how to represent knowledge. Examples include decision trees, sets of rules, instances, graphical models, neural networks, support vector machines, model ensembles and others.
- **Evaluation**: the way to evaluate candidate programs (hypotheses). Examples include accuracy, prediction and recall, squared error, likelihood, posterior probability, cost, margin, entropy, k-L divergence and others.

Representation



Size of the data set



- For a given model complexity, the over-fitting problem become less severe as the size of the data set increases.
- The larger the data set, the more flexible is the model that we can afford to fit to the data.
- One rough heuristic that is sometimes advocated is that the number of data points should be no less than some multiple (say 5 or 10) of the number of adaptive parameters in the model.

Inductive Learning

- Inductive Learning is where we are given examples as data (x) and the output of the function ($f(x)$). The goal of inductive learning is to learn the function for new data (x).

- **Classification**: when the function being learned is discrete.
- **Regression**: when the function being learned is continuous.
- **Probability Estimation**: when the output of the function is a probability.

Performance Measure

- Classification Error
- Prediction Error
- Cluster quality
- Support and Confidence for Association rule
- Cost for Reinforecement Learning