INF8953CE - Fall 2020

Machine Learning

- Sarath Chandar

1. Introduction to Machine Learning

1. Introduction to Machine Learning

Study of algorithms that

- improve their performance P
- at some task T
- with experience E

Learning task : $\langle P, T, E \rangle$

Example :-

1) T : Spam classification

P : Accuracy

E : Set of example mails labelled as spam or not spam.

2) T : Credit card fraudulent transaction detection

P : Accuracy

E : Set of historical transactions marked as legit or fraud.

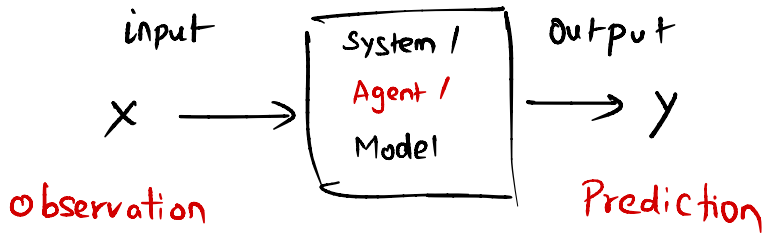
3) T : Playing the game of chess

P : number of games won

E : Set of game trajectories from expert players.

Prediction :

The most common ML application.



Examples:-

① Given the size of the house, predict the selling price.

x : Size of the house.

y : Selling price.

② Given the current stock price of a company, predict the same after 10 minutes.

x : Stock price at time t

y : Stock price at time t+10

③ Given an image, predict the object in the image.

x : Image as pixels

y : object name from a predefined set

e.g. { dog, cat, birds, flight, ball }

④ Given an image of a handwritten digit, predict the digit.

x: 

y: ∈ [one of {0, 1, 2.. 9}]

⑤ Given the temperature, humidity, wind predict whether it will rain or not.

x: temp, humidity, wind

y: Yes / No

Note: ① In examples ① and ②, y is a real number.

$y \in \mathbb{R}$

② In examples ③, ④, ⑤, y is categorical.

$y \in$ {dog, cat, birds, flight, ball} in ③

$y \in \{0, 1, \ldots 9\}$ in ④

$y \in$ {yes, no} in ⑤

$y \in \mathbb{R} \Rightarrow$ regression problem.

$y \in \{c_1, \ldots c_n\} \Rightarrow$ classification problem

↑ Categorical

In ⑤, $y \in \{yes, no\}$ → binary classification problem.

Prediction task :- Given some input x, make a good prediction of output y, denoted by $\hat{y}$.

x can be a vector $x = \langle x_1, x_2 \ldots x_p \rangle$

↳ 'p' features

y can be a scalar or a vector.

Note: y → target

$\hat{y}$ → Model's prediction

Supervised Learning :-

Given a training set $\{x^{(i)}, y^{(i)}\}_{i=1}^{N}$ of N datapoints, learn a prediction function $f : x \to y$ such that given a new x, f can accurately predict the corresponding y.

Note: The prediction function `f` is useful only if it can make accurate predictions for unseen `x`. We will call this capability as generalization to unseen instances. Generalization is a key requirement for any ML algorithm.

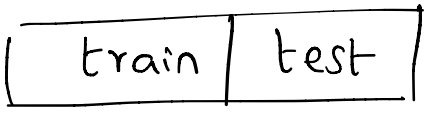
Simple example: Housing price prediction in Portland, Oregon.

Jhon, who lives in Portland, Oregon wants to sell his house and wants to know what a good market price would be. One way to do this is to first collect information on recent houses sold and make a model of housing prices. This is an example for regression.

Let us consider the following dataset :

X Size of the house (in square feet)	Y Price of the house.
2104	399,900
1600	329,900
2400	369,000
⋮	⋮

Let us say we have 47 datapoints. The first step would be split the data into training data and test data.



(1) We will use the training data to train the model.

(2) We will use the test data

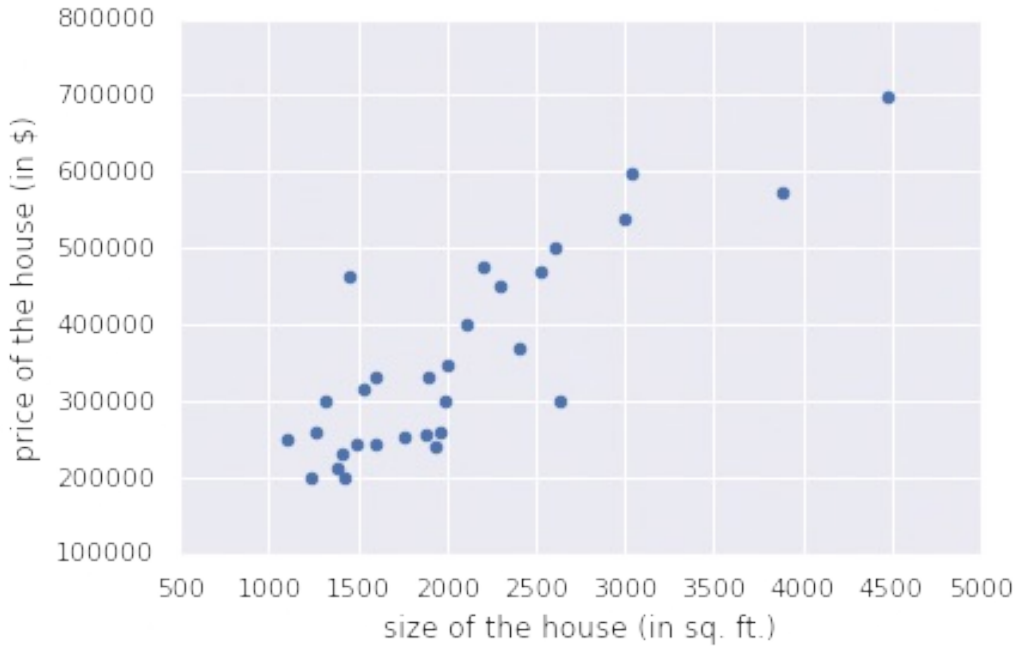
only to test the generalization of the model.

47 30 17

Note: Test set is a proxy for the true performance of the model when it sees a new 'x' after training.

Let us divide the dataset into 30 training instances and 17 test instances.

Visualization of training data:-



y

x

Now consider this simple model:

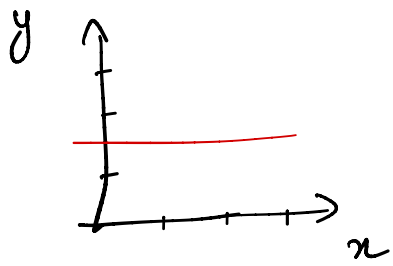
$$\hat{y} = w_0 + w_1 x$$

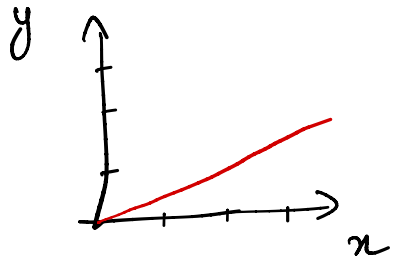
this is the equation of a line, we can tweek it to get a line to fit the data better

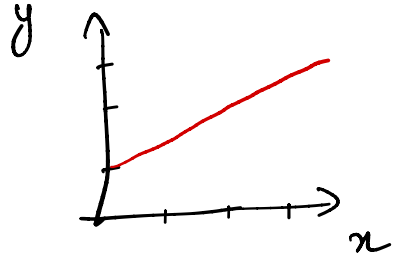
w_0 → y-intercept

What kind of curves can this model fit?

This model can only fit lines.







w0--> y intercepts

$w_0 = 1.5$ $\quad w_1 = 0$

$w_0 = 0$ $\quad w_1 = 0.5$

$w_0 = 1$ $\quad w_1 = 0.5$

→ Note the significance of adding w_0.

→ If you dont add w_0, you can only cover lines passing through the origin!

w_0 = bias in ML literature.

$$\hat{y} = w_0 + w_1 x$$

Parameters of the model.

$$w = (w_0, w_1)$$

i/p, Parameter

$$\hat{y}(x; w) = w_0 + w_1 x$$

How can we learn the parameters w_0, w_1 ?

First step is to define an objective function that the model should achieve.

Objective: we want $\hat{y}(x; w)$ to be as close as possible to y.

This can be done by minimizing the following error function

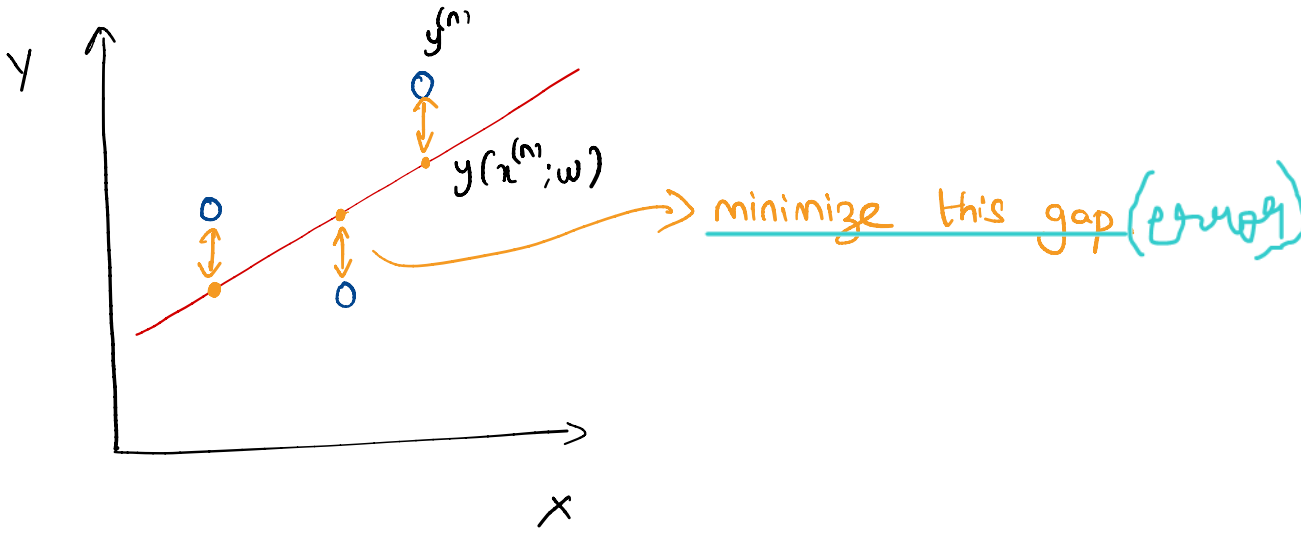
Number of training instances.

$$E(\omega) = \frac{1}{2} \sum_{n=1}^{N} \left\{ \hat{y}(x^{(n)}; \omega) - y^{(n)} \right\}^2$$

least squares error function.

model's prediction

target.

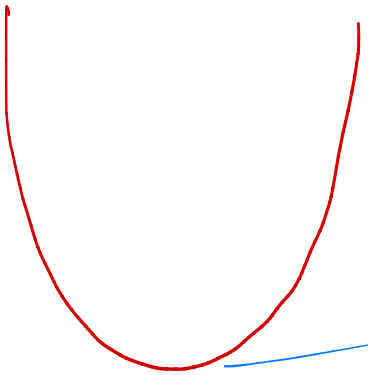


$$E(\omega) = \frac{1}{2} \sum_{n=1}^{N} \left\{ (\omega_0 + \omega_1 x^{(n)}) - y^{(n)} \right\}^2$$

objective : $\min_{\omega} E(\omega)$

How to find the minimum?

→ This error fn. is a quadratic fn. of the parameters.



→ there is only one minimum.

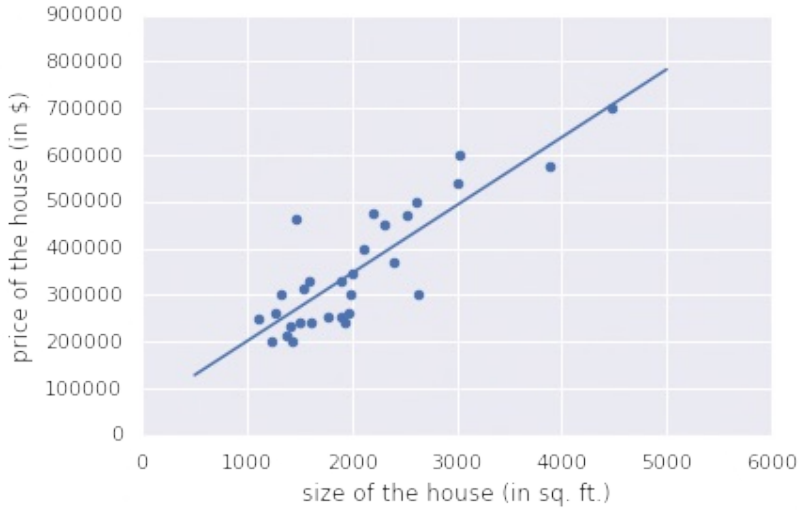
At the minimum, the derivative of the fn. w.r.t. the parameters w will be zero.

i.e. $$\frac{dE}{dw} = 0.$$

↳ derivatives of $E(w)$ w.r.t. w will be linear in the elements of w.

↳ min $E(w)$ has a unique solution w^* which can be found in closed form.

The fitted model:



$$\hat{y}(x; w) = w_0 + w_1 x$$

Note: This is an example of a linear model.

Linear model: The model is linear in terms of the parameters.

Linear models with quadratic error functions have an unique closed form solution!

→ More on this later.

Switching to vector/Matrix notation:

$$\hat{y} = w_0 + w_1 x$$

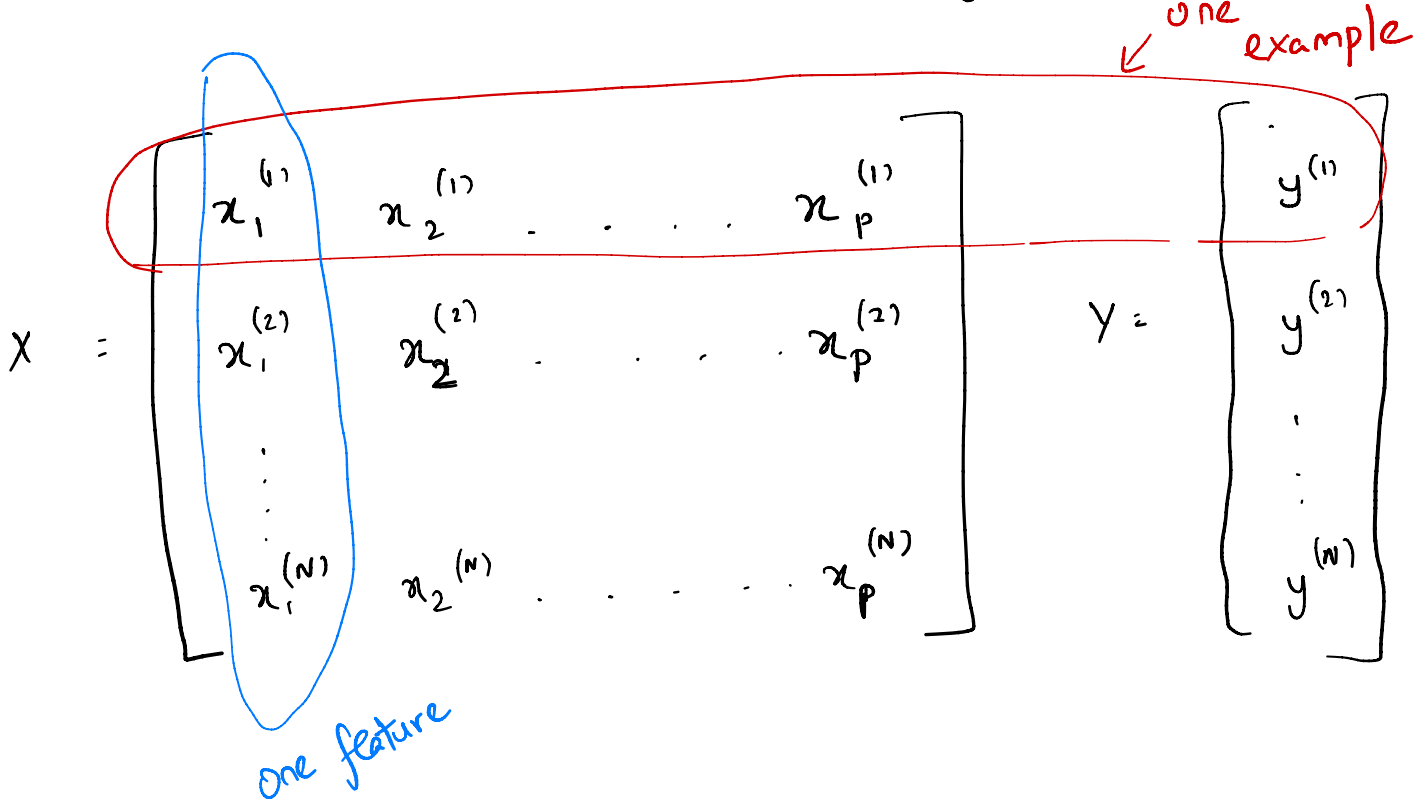
$$\hat{y} = w_0 \cdot 1 + w_1 x$$

Let $w = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$, $x = \begin{pmatrix} 1 \\ x \end{pmatrix}$

Then $\hat{y} = w^T x$.

Note: We will often prepend x with $\underline{1}$ to avoid treating the bias separately.

Now consider the entire training data:



X = data matrix y = target vector.

rows in X = examples.

Columns in X = features.

X = $N \times P$ matrix.

y = $N \times 1$ vector

w = $P \times 1$ vector.

$$\hat{y} = X w$$

$N \times P$ $P \times 1$ → $N \times 1$

Single matrix vector multiplication to predict y for all N examples.

Vectorizing will increase the speed compared with for loop

$$E(w) = \frac{1}{2}(y - Xw)^T (y - Xw)$$

$\frac{1}{2}$: 1×1 (scalar)

$(y - Xw)^T$: $N \times 1$ → $1 \times N$

$(y - Xw)$: $N \times 1$

$N \times P$ $P \times 1$ → $N \times 1$

$1 \times N$ $N \times 1$ → 1 scalar.

Solution to least squares:-

$$E(w) = \frac{1}{2}(y - xw)^T (y - xw)$$

diff. w.r.t. w, set it to zero to find the minimum.

$$-\frac{2}{2} x^T (y - xw) = 0$$

$$x^T (y - xw) = 0$$

$$x^T y - x^T x w = 0$$

$$x^T x w = x^T y$$

$$\boxed{w^* = (x^T x)^{-1} x^T y}$$

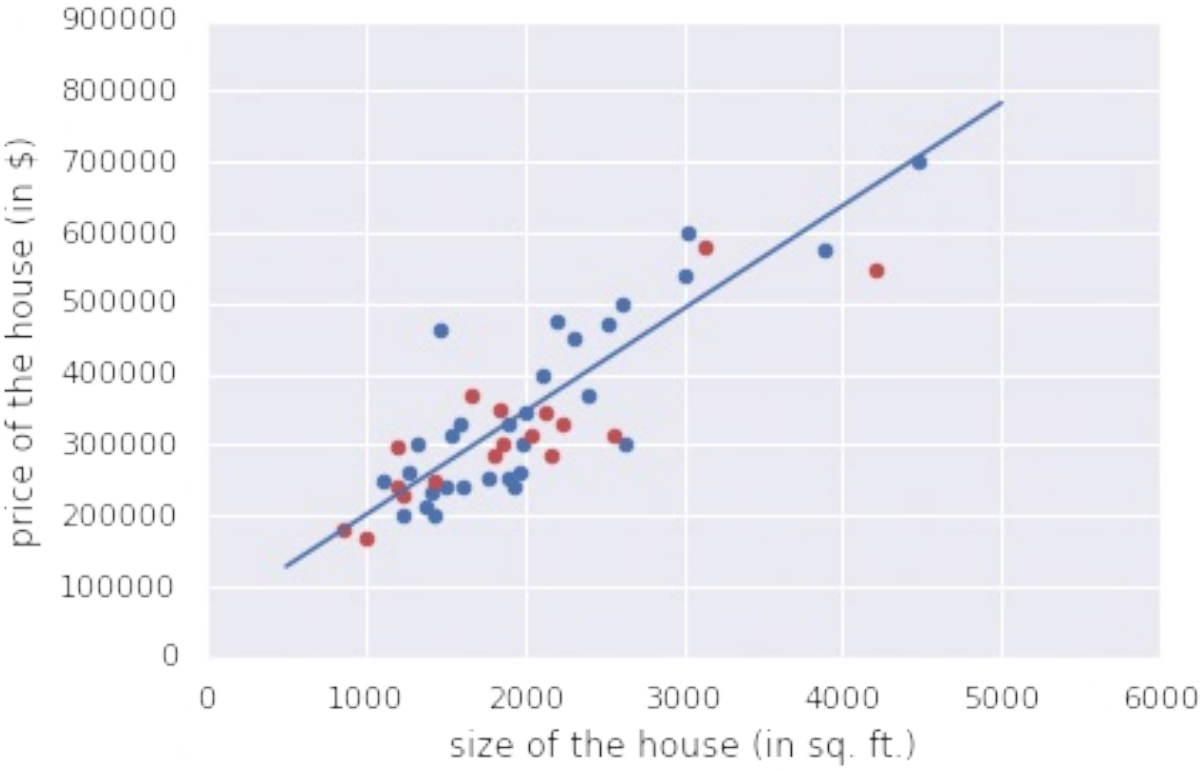
Moore - Penrose Pseudo-inverse of matrix x

Now given a new x, we can find the target y as follows:

$$\hat{y} = w^{*T} x.$$

Back to housing price prediction example:-

How good is the learnt linear regression model?



Least Squares error (LSE)

$$E(\omega) = \frac{1}{2} \sum_{i=1}^{N} \left(y^{(i)} - \omega^T x^{(i)} \right)^2$$

← Not in same scale as prediction

↑ depends on N.

if you have small data, you have small error.
if you have large data, you will get large error

Better metric for performance :

Root Mean Square Error (RMSE)

to make the error rate robust ie, if there datapoints is more, then the error will also be hence,it is handled

$$E_{RMS} = \sqrt{\frac{2\,E(w)}{N}} \rightarrow \text{AVG}$$

Note : RMSE is on the same scale as y.

Train RMSE = 68727.04

Test RMSE = 57976.80.

The model is off by approx. 50k $. still a decent performance for a simple model.

Summary :

Linear regression

Model : $\hat{y} = w^T x$ (linear model)

Parameters : w

objective fn : $E(w) = \frac{1}{2}(Y - Xw)^T (Y - Xw)$

Solution : $w^* = (x^T x)^{-1} x^T y$

Metric : $RMSE = \sqrt{2E(w)/N}$

Synthetic example:

Consider the following synthetic function

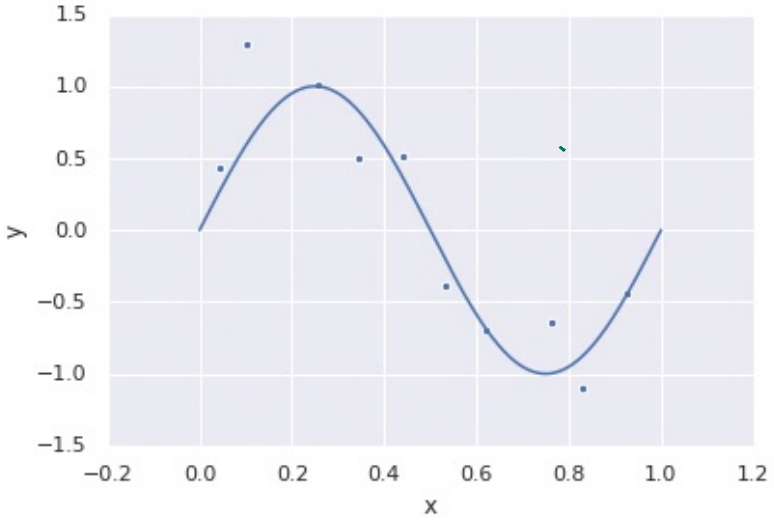
$$y = \sin(2\pi x) + \epsilon$$

where we have added a small Gaussian noise ϵ to the output of the sin function.

$$\epsilon \sim \mathcal{N}(0, 0.3)$$

↑ mean ↑ std. deviation

The training data consists of the following 10 datapoints sampled from this fn.

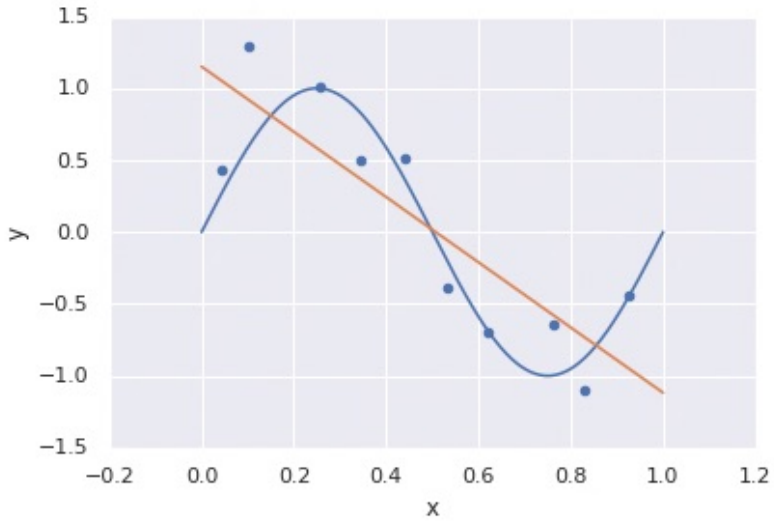


Note 1: In real life problems, we will not have access to true function y.

Note 2: Adding Gaussian noise to the sin fn is reasonable. Because,

even in real life, observed 'y' will have some noise due to observation errors or the inherent stochasticity of the process that generates the function 'y'.

Linear regression solution:-



Looks like the learnt line fn does not fit the data very well.

We can consider higher order polynomials!

$$\hat{y}(x; w) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$

Note: x^2 : x raised to the power of 2.

$x^{(2)}$: x of the second example

x_2 : 2nd feature of x.

Using the fact that $x^0 = 1$, we can write

$$\hat{y}(x;w) = w_0 x^0 + w_1 x^1 + \ldots + w_M x^M$$

$$= \sum_{j=0}^{M} w_j x^j$$

Let $w = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{pmatrix}$ $\quad x = \begin{pmatrix} x^0 \\ x^1 \\ \vdots \\ x^M \end{pmatrix}$

Although polynomial regression fits a nonlinear model to the data, as a statistical estimation problem it is linear

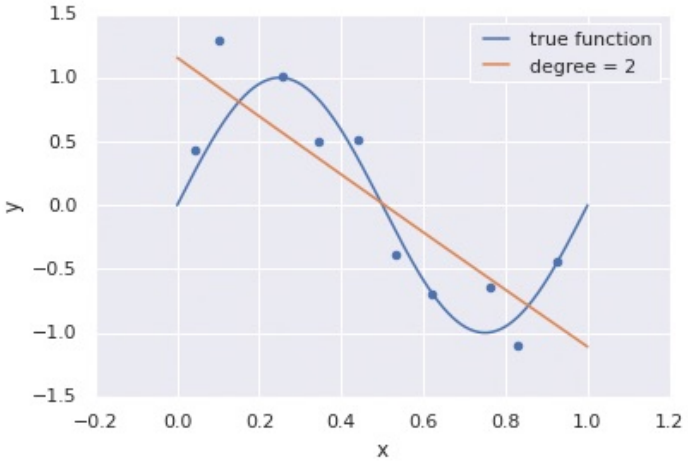
$$\hat{y}(x;w) = w^T x.$$

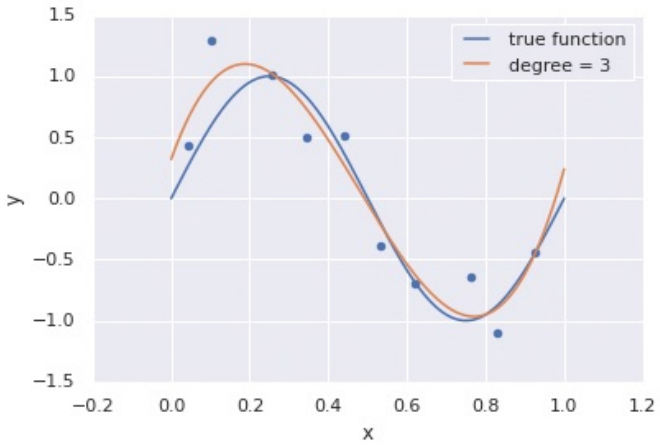
Note: $\hat{y}(x;w)$ is a non-linear fn. of x. But $\hat{y}(x;w)$ is still a linear fn. of w! So this is still a linear model.

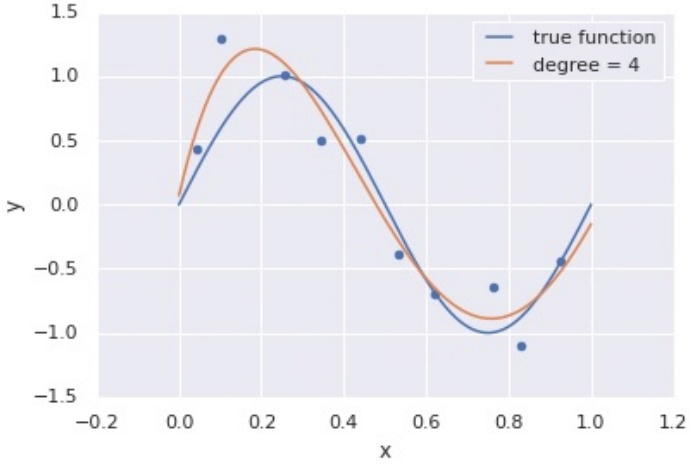
$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \left\{ \hat{y}(x^{(n)};w) - y^n \right\}^2$$

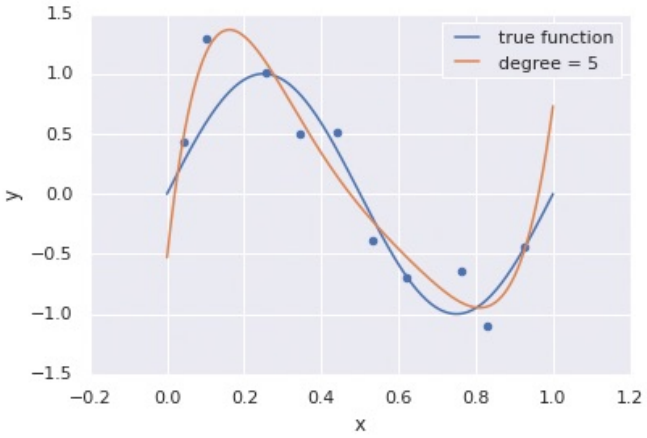
is still a quadratic fn. of w. Hence, unique solution exists.

x is a vector instead of being a scalar. However, we can still use the same least squares solution.







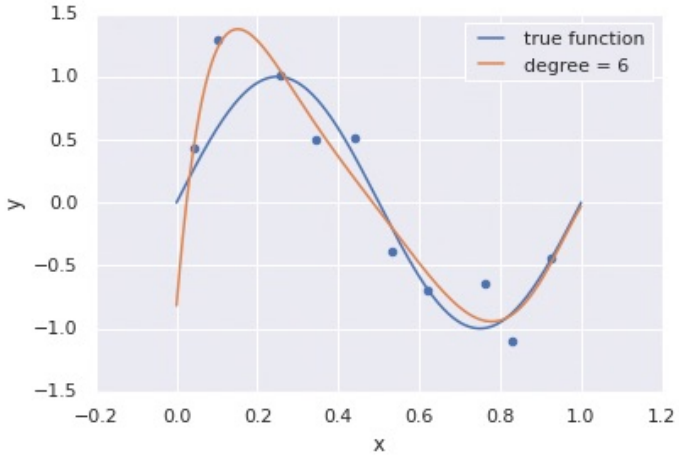


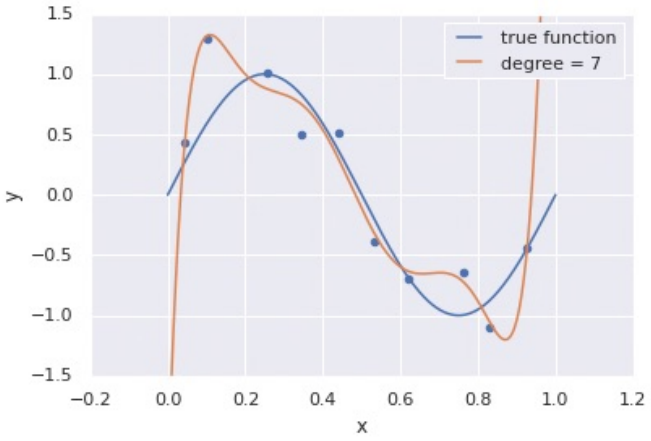
$M = 2$ is still not good.

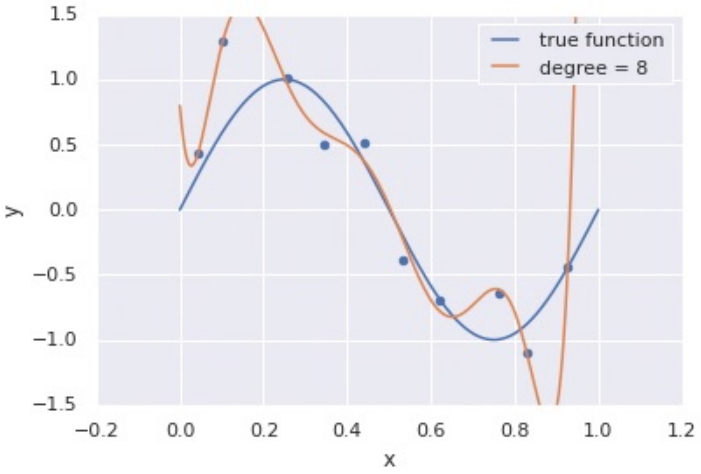
Solutions for $M = 3, 4$ looks good.

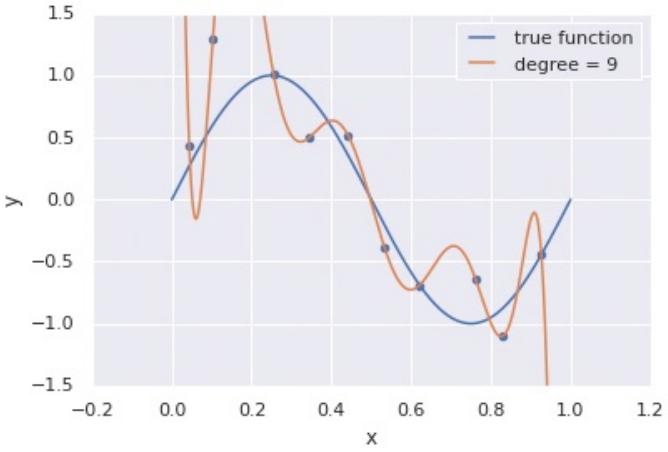
Solution for $M = 5$ is slightly deviating from the Sin curve.

What happens as we keep increasing M?









Solution for $M = 9$ achieves zero training error!

Is this a good solution?

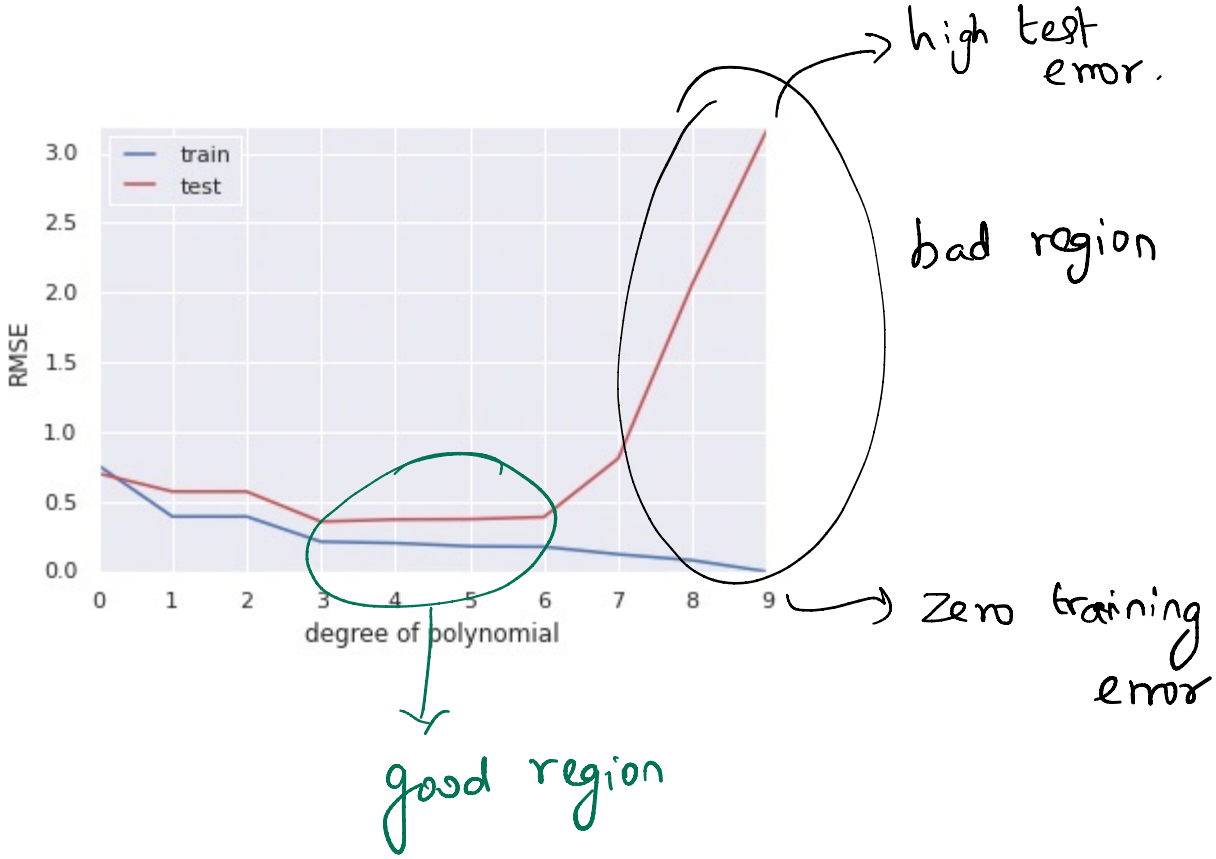
NO. The fitted curve oscillates wildly and gives a poor representation of the fn $\sin(2\pi x)$.

↳ This is known as overfitting.

In this example, we know the true function. So we can tell that $M=9$ is not a good approximation of it.

How to tell if a model is overfitting when we dont know the true function?

→ If a model is overfitting, its generalization performance should be bad. i.e. the test error should be high.



Wait! 9 degree polynomial contains 3 degree polynomial as special case. Then why M = 9 perfoms bad?

Let us look at the learnt w^*.

	M=0	M=1	M=3	M=6	M=9
w_0^*	0.05	1.15	0.32	-0.82	13.64
w_1^*	0.0	-2.27	9.04	38.62	-637.4
w_2^*	0.0	0.0	-29.95	-240.94	10762.47
w_3^*	0.0	0.0	20.83	674.04	-85516.44
w_4^*	0.0	0.0	0.0	-1001.87	375773.12
w_5^*	0.0	0.0	0.0	751.69	-984759.25
w_6^*	0.0	0.0	0.0	-220.76	1576883.5
w_7^*	0.0	0.0	0.0	0.0	-1512453.8
w_8^*	0.0	0.0	0.0	0.0	797880.06
w_9^*	0.0	0.0	0.0	0.0	-177972.81

As M increases, the magnitude of the coeff. gets larger.

For M=9, 10 weights are heavily tuned for the given 10 datapoints!

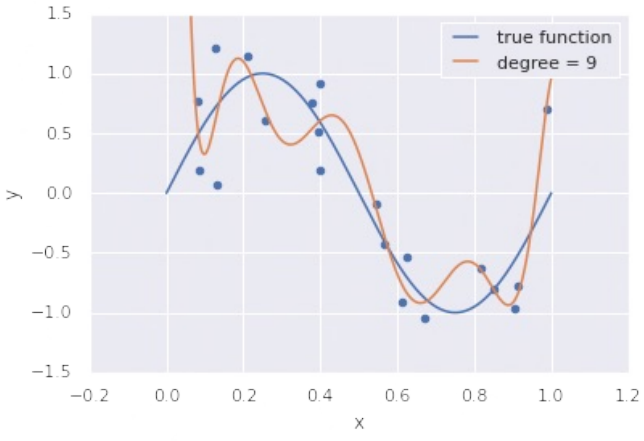
9-degree polynomial contains the 3-degree polynomial. So a 9-degree model should be able to recover the 3-degree Solution by setting the remaining weights to 0.

In other words, 9-degree polynomial model is expressive enough to model the given sin function.

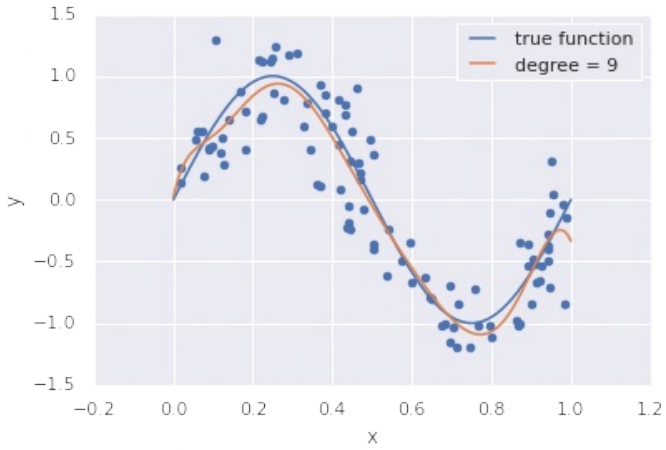
But we are not able to learn that solution.

How to solve this issue of overfitting ?

Solution 1: Add more data points / examples so that the model cannot overfit.



15 examples.



100 examples.

With 100 datapoints, M=9 model approximates the true function very well!

What if you cannot obtain more data?

Solution 2:- Add a penalty term to the error function in order to discourage the coefficients from reaching large values.

ex: $\tilde{E}(w) = \frac{1}{2}\sum_{n=1}^{N}\{\hat{y}(x^{(n)};w) - y^{(n)}\}^2$

$$+\frac{\lambda}{2}\|w\|^2$$

where $\|w\|^2 = w^T w = w_0^2 + w_1^2 + \cdots + w_M^2$

This is known as regularization. regularization term.

λ = Controls the relative importance of the regulalization term.

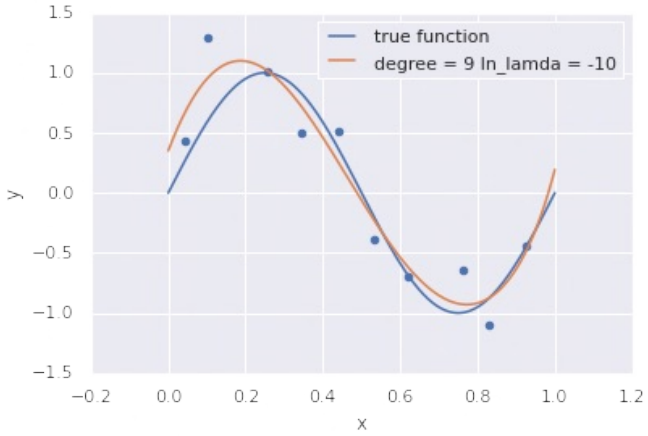
Note: $\tilde{E}(w)$ is still a quadratic fn. of w.

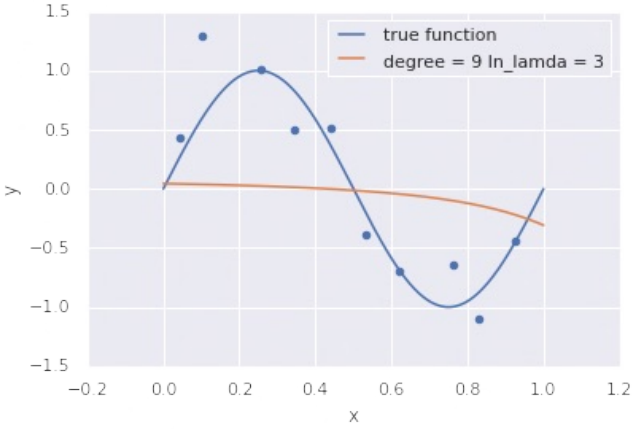
This error fn. can be minimized exactly in closed form.

We will derive this solution later.

How to choose λ?

It is easy to work with $\ln \lambda$ and Compute λ as $e^{\ln \lambda}$.





	$\ln \lambda = -\infty$	$\ln \lambda = -10$	$\ln \lambda = 3$
w_0^*	13.64	0.35	0.04
w_1^*	-637.4	8.49	-0.06
w_2^*	10762.47	-26.14	-0.06
w_3^*	-85516.44	9.11	-0.05
w_4^*	375773.12	11.85	-0.04
w_5^*	-984759.25	3.16	-0.04
w_6^*	1576883.5	-3.88	-0.03
w_7^*	-1512453.8	-5.68	-0.02
w_8^*	797880.06	-2.27	-0.02
w_9^*	-177972.81	5.19	-0.02

↑ No regularization

↑ too much regularization.

When λ is too small, then no regularization.

When λ is too high, too much regularization.

It is very crucial to choose the right λ.

λ = hyper-parameter of the model.

We will normally fix the hyper-parameter and learn the parameters from the data.

But how to choose λ? Can we use the test set to choose λ?

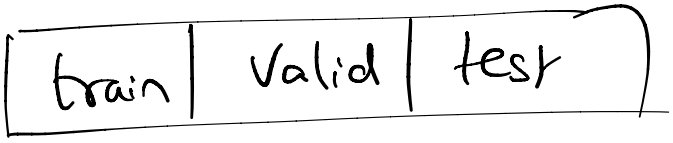
No

Test set is supposed to be a proxy for completely new x.

If you choose λ based on the test set, then the model has seen the test set. So the test set will not be the proxy for the real performance of the model.

What else can we do?

Create a separate hold out set.



① For different values of λ,

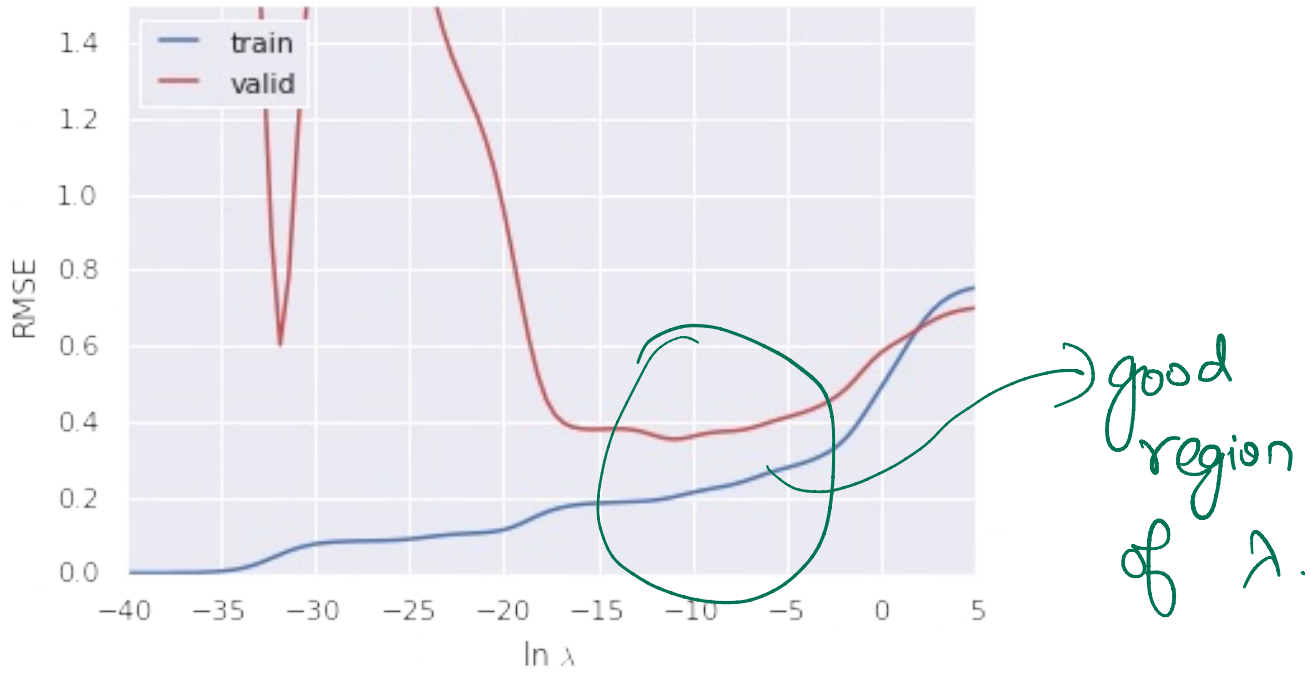
✓ train the model.

✓ Compute the performance in valid set.

② Pick the value of λ that has best validation performance.

③ Compute the test performance for the model with chosen λ.

This procedure is known as model selection.



In this example, we knew the true function is a 3-degree polynomial and hence we can obtain it from the 9-degree polynomial by regularization.

What if the true function was a 15-degree polynomial and we dont know that?

In other words, How can we choose the value of M?

M - is also a hyperparameter!

Solution 3:- Try different values of M and select the value of M based on the performance in val. set.

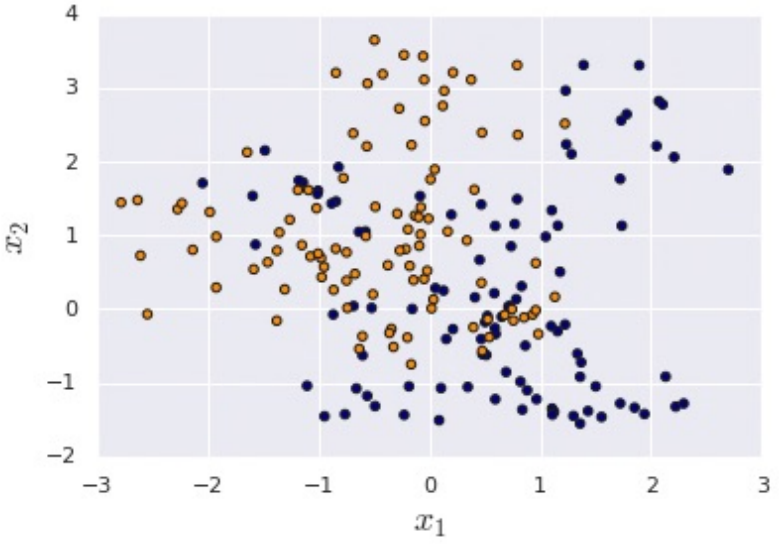
This is also model selection!

Machine Learning Pipeline :-

1. Define the input and output : x, y
2. Collect examples for the task.
3. Divide the examples into train / valid / test set.
4. Do data preprocessing.
4. Define your model
   - Model will consist of Parameters and hyperparameters.
5. Define the error fn or loss fn you want to minimize.
6. For different values of hyper-parameters,
   - learn model params by minimizing the loss fn.
   - Compute validation performance.
7. Pick the best model based on the val. perf.
8. Test the model with test set.

Classification :-

Consider the following binary classification problem. We are interested in classifying the data points as either "blue" class or "Orange" class.



⇒ There are 2 features : x_1 & x_2.

⇒ We have 100 examples per class.

Class 1 : blue

class 2 : Orange.

We will always convert the targets to numbers.

Blue = 0
Orange = 1

(or)

Blue = −1
Orange = +1

Solution 1: Linear model

We can use the same idea from regression.

For $\{0,1\}$ classification,

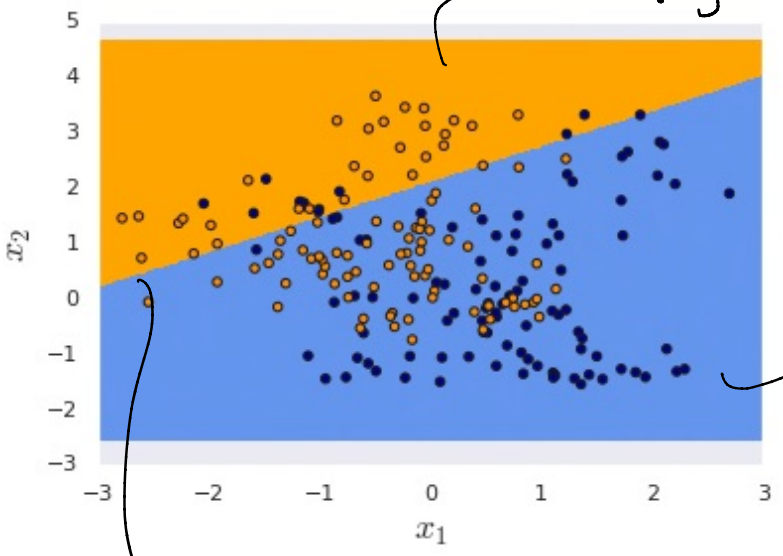
$$\hat{y} = \begin{cases} 1 & \text{if } \hat{y} > 0.5 \\ 0 & \text{if } \hat{y} \leq 0.5 \end{cases}$$

$\hat{y} = 0.5$ Decision boundary.

For $\{-1,1\}$ classification,

$$\hat{y} = \begin{cases} -1 & \text{if } \hat{y} < 0 \\ 1 & \text{if } \hat{y} \geqslant 0 \end{cases}$$

$\hat{y} = 0$ Decision boundary.



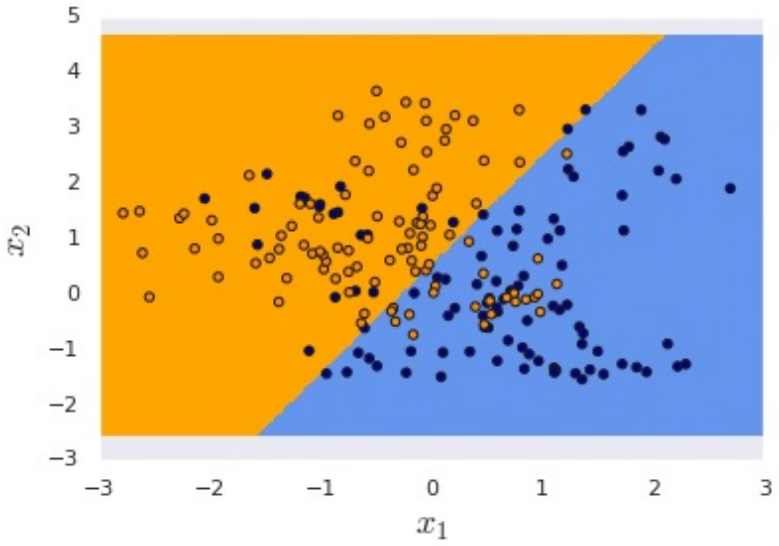
orange region

blue region

decision boundary

Linear model without bias term.

$w^T x = 0.5$



Linear model with bias term.

Note: Some blues wrongly classified as orange & vice-versa.

Clearly line is not a good decision boundary for this classification problem.

Solution 2: Nearest-neighbor methods.

Use those observations in the training set $\mathcal{T}$ closest in input space to x to form $\hat{y}$.

If more neighbors are class 0, then predict class 0 and vice-versa.

$$\hat{y}(x) = \frac{1}{k} \sum_{x^{(i)} \in N_k(x)} y^{(i)}$$

$N_k(x)$ = Neighborhood of x.

↳ k closest points $x^{(i)}$ in T.

↓

what metric ?

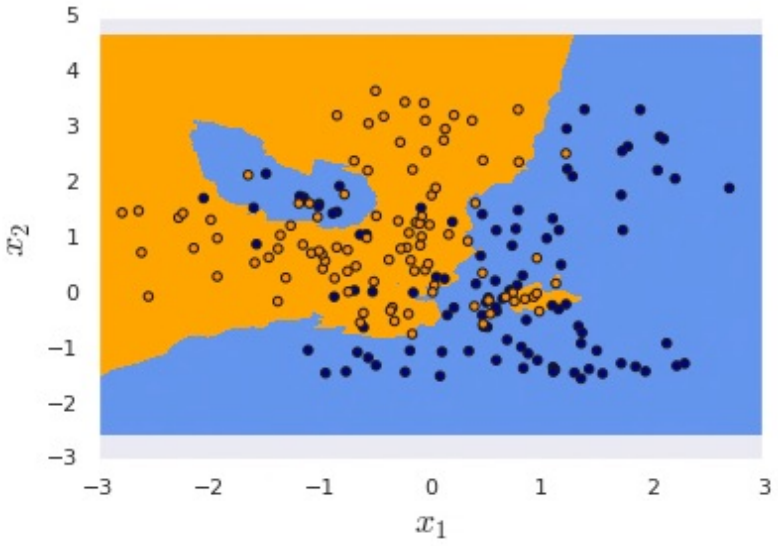
Euclidean distance.

if $\hat{y}(x) > 0.5$, class 1

else class 0.

This is like majority voting from the neighbors.

This model assumes that the class distribution is locally smooth.

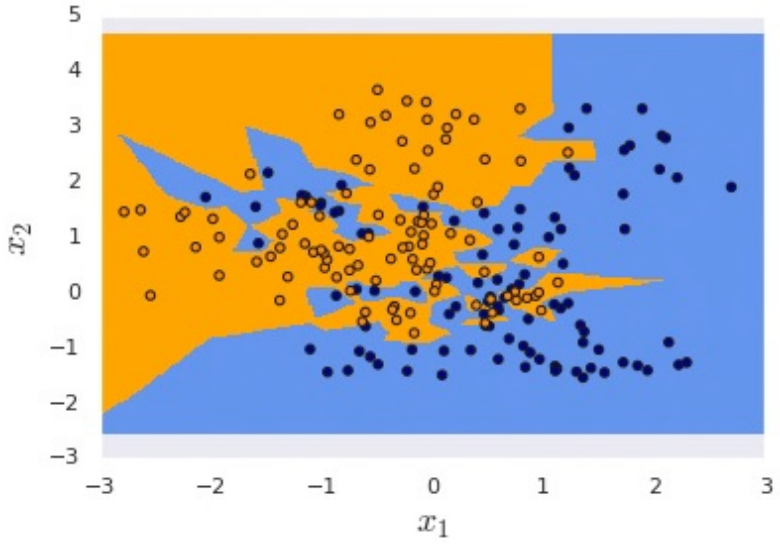


$K = 10$

decision boundary is far more irregular and responds to local clusters where one class dominates.

There are still some misclassifications.

1-NN



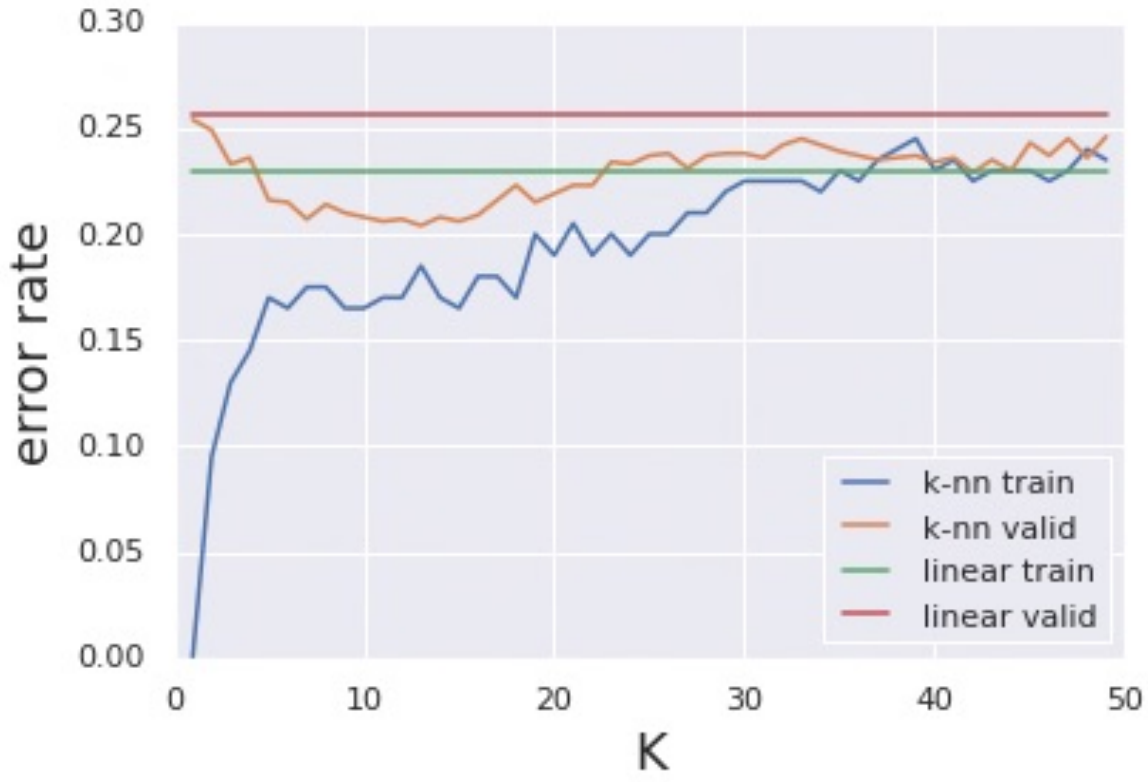
$K = 1$.

The decision boundary is even more irregular than before.

$\Rightarrow$ There are no misclassification.

K = hyper-parameter of the nearest neighbor algorithm.

Let us do model selection using a separate validation set.



k=1 overfits to the training data!

k-NN performs better than linear model.

↑
Non-linear model.

Are there any other hyperparam. for k-NN algorithm?

Metric to Compute N.N.

Least squares Vs. Nearest neighbors:-

Least squares	Nearest Neighbors.
- Decision boundary is very smooth.	- Decision boundary is wiggly and depends on a handful of input points & their positions.
- More stable	- Less stable.
- Assumes that the decision boundary is linear. ↳ strong assumption	- Assumes that the class distribution is locally smooth.
- high bias, low variance	- low bias, high Variance.

You should know!

1. Prediction problem.
2. Regression / classification
3. Supervised learning / Generalization
4. Linear regression
5. bias in linear model.
6. Parameter / hyper parameter of a model.
7. Overfitting
8. Solutions to overfitting
   - Add more data
   - Regularization.
9. Model selection
10. ML pipeline
11. Nearest neighbor classifier.