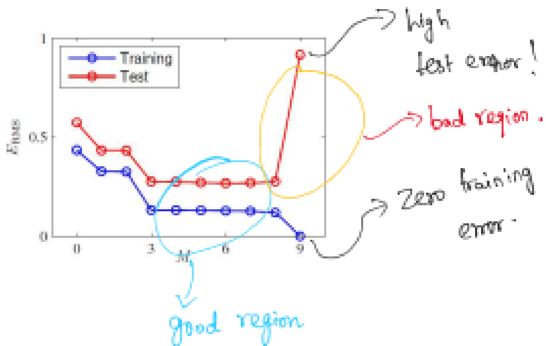
Continuation from last lecture:

For M=9, we obtain excellent fit to the training data. The fitted curve oscillated wildly and is a poor representation of $sin(2\pi x)$.
$\implies$ overfitting.

> What do we do when you dont know the true function ? Use a separate test test. use most of data for training and rest for testing.

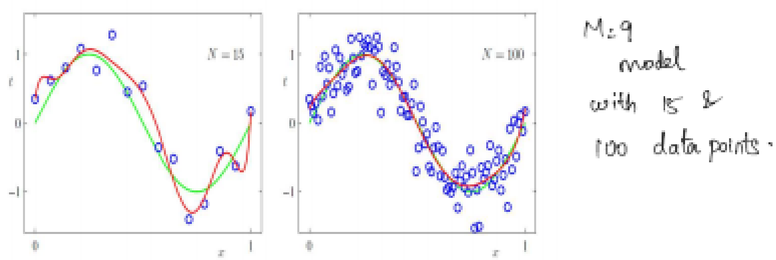
Root mean square error:

$$E_{RMS} = \sqrt{\frac{2E^*(w)}{N}}$$



How to Solve overfitting ?

Solution 1: Add more data points to get a better training performance.



But what if we can't add more data points?

Solution 2: Add a penalty term to the error function, In order to discourage the coeff from reaching large values.

$$E(w) = \frac{1}{2}\Sigma_{n=1}^{N}\{\hat{y}(x^{(n)}; w) - y^{(n)}\}^2 + \frac{\lambda}{2}\|w\|^2$$

When λ is too small, there is no regularization, if λ is too high, there is big regularization. It is crucial to choose a correct λ (hyper-Parameter). Normally we fix the hyper-parameter, and then we learn the parameter.

How to choose λ? can we choose the test set to choose λ?
First, we answer the 2nd question, No, the test set is supposed to be to find the new corresponding y for a new x, bc otherwise, λ has seen the test data.
We need a seperate hold-out set. bascially, we devide the data points into, train, valid, test.

1. for different calues of λ: train the model and then compute valid performance.
2. Pick the value of λ that has the best validation performance.
3. Compute the test performance for the model with chosen λ.

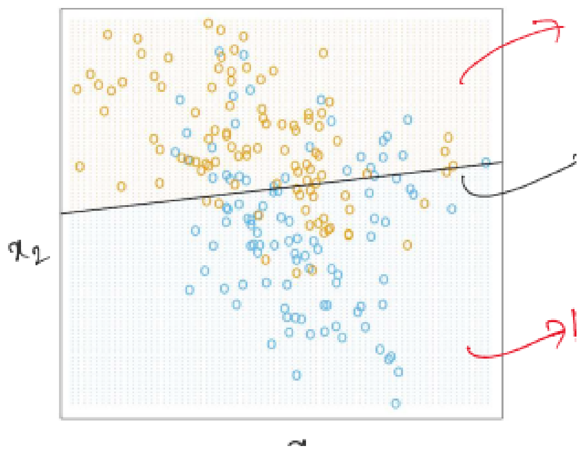
$\Longrightarrow$ **Model selection**.

In our case, we knew the original function. but what if the true fn is 15 deg poly and we start with M=9? M is also a hyper-parameter.
solution3: Try different values of M and select the value of M based on Validation performance. This is also **Model selection**.

Machine Learning Pipeline:

1. Define the input and output: xy.
2. Collect examples for the task.
3. Divide the examples into train/valid/test sets.
4. Define your model: parameters, hyper-parameters
5. Define the error fn/loss fn you want to minimize.
6. For different values of hyper-parameter.
   - learn model param by min.loss fn
   - compute validation performance
7. Pick the best model based on validation performance.
8. test the model with test set.

Now we're going to attempt a Classification problem:

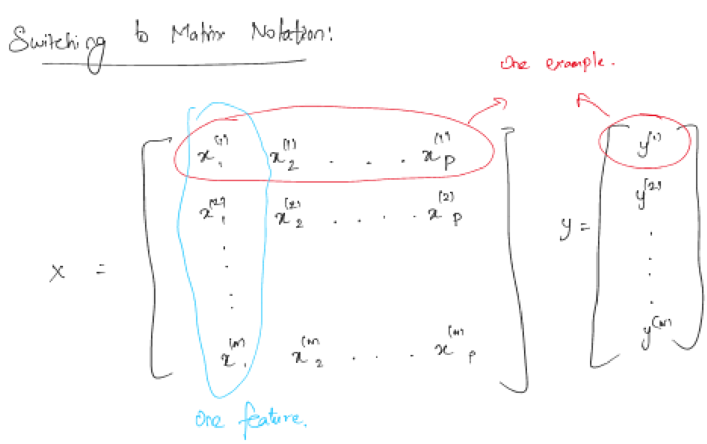


we will always convert the target to numbers, e.g blue =0, orange = 1.

Model 1: Linear model.
$x = (x_1, x_2)$. $x^T = (x_1, x_2, .., x_p)$.
$\hat{y} = \Sigma_{j=0}^{p} w_j x^j = x^T w$.

$$RSS(w) = \frac{1}{2}\Sigma_{n=1}^{N}\{y^{(n)} - x^{(n)T}w\}^2$$

We switch into Matrix notation.



rows in X: examples, cols in X: Features. X = NxP matrix y = N-vector, w = p-vector.

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

.

Soluion: Differentiate with respect to w and solve for 0 to find the minimum.

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

$$x^T y - x^T x w = 0$$
$$w^* = (x^T x)^{-1} x^T y$$

now given a new x, we find a new y using the following:

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

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

Model 2: Nearest-neighber methods.
Use those observations in the training set T closest in input space to x to from $\hat{y}$.

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

$N_k(x)$-neighberhood of x, closet k points $x^{(i)}$. What metric ? Euclidean distance.

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

This model assumes that the class distribution is locally smooth.