COMP [56]630– Machine Learning

Lecture 3 – Linear Regression Pt. 1

Logistics

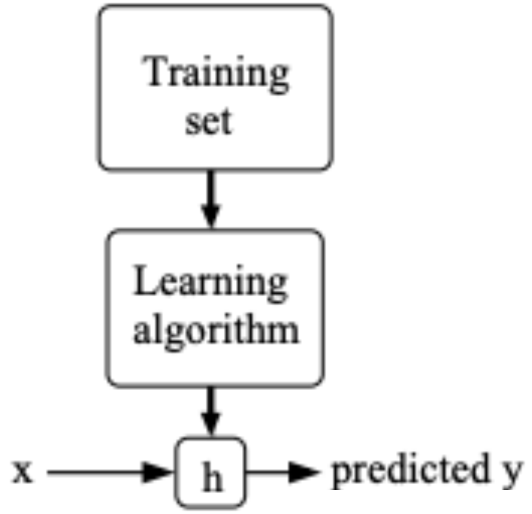
- Assignment 0 due Friday 01/26 11:59 pm
- No Quiz this week

Supervised Learning – Linear Models

Finally!

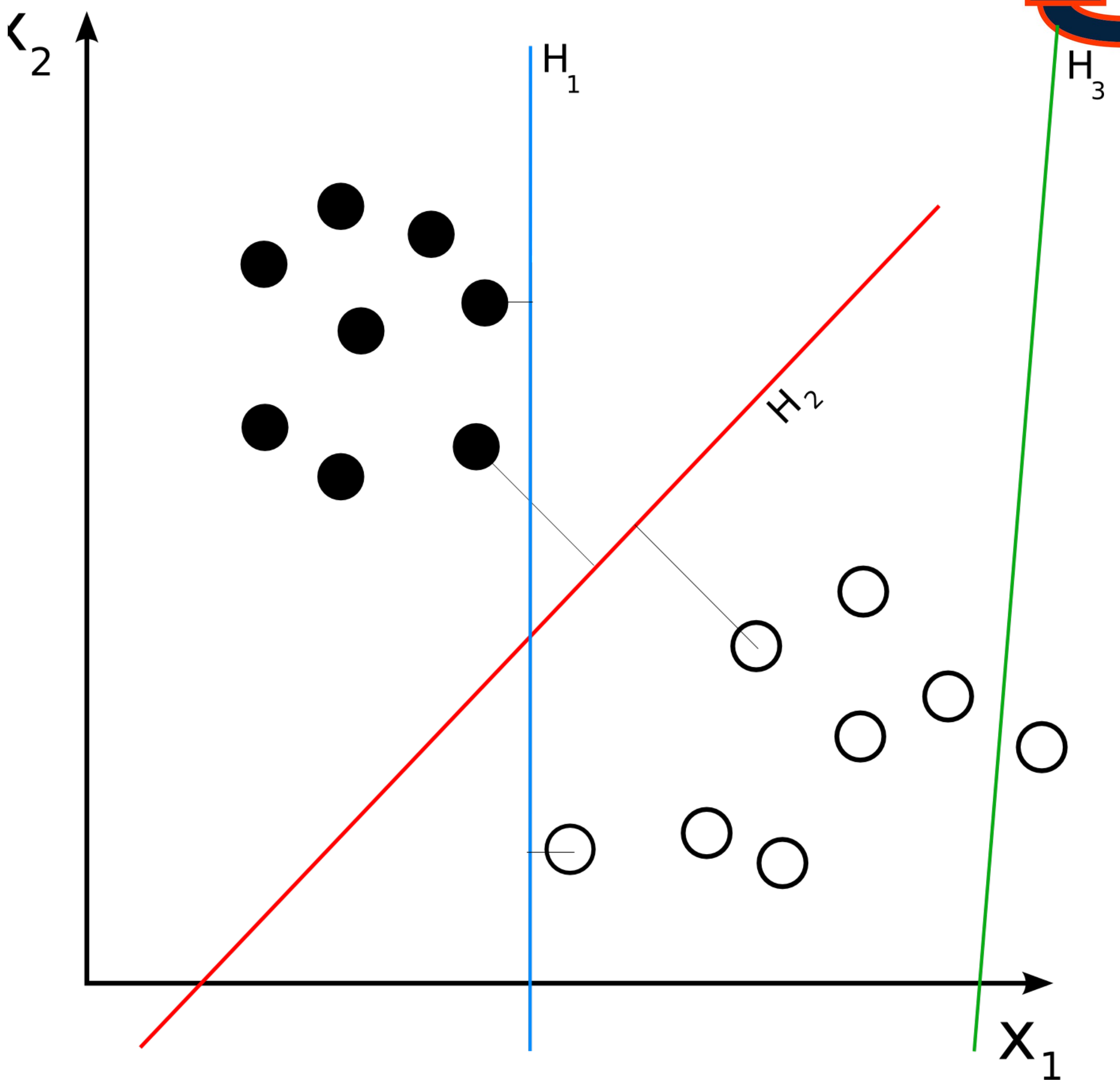
The basics

- Input: a set of inputs X = {x_1, x_2, … x_n}, also called ***features***
- Output: a set of expected outputs or ***targets*** Y = {y_1, y_2, … y_n}
- Goal: to learn a function $h : X \rightarrow Y$ such that the function $h(x_i)$ is a good predictor of the corresponding value y_i
  - h(x) is called the ***hypothesis***
- If the target is continuous the problem setting is called ***regression.***
- If the target is discrete or categorical, the problem is called ***classification.***



Linear Classifier

- The simplest ML model
- Makes a *classification* decision based on the value of ***a linear combination*** of the characteristics (features).
- Black and white circles are different labels. H_1, H_2, … represent different *decision boundaries* i.e. linear functions that best map the classification process.
  - ***Goal:*** find the best linear function that has highest accuracy



Linear Classifier (cntd.)

- Mathematically represented as

$$y = \boldsymbol{W}\boldsymbol{x}^{\boldsymbol{T}} + b$$

where y → labels (vector)

W → model parameter matrix

x → feature vector

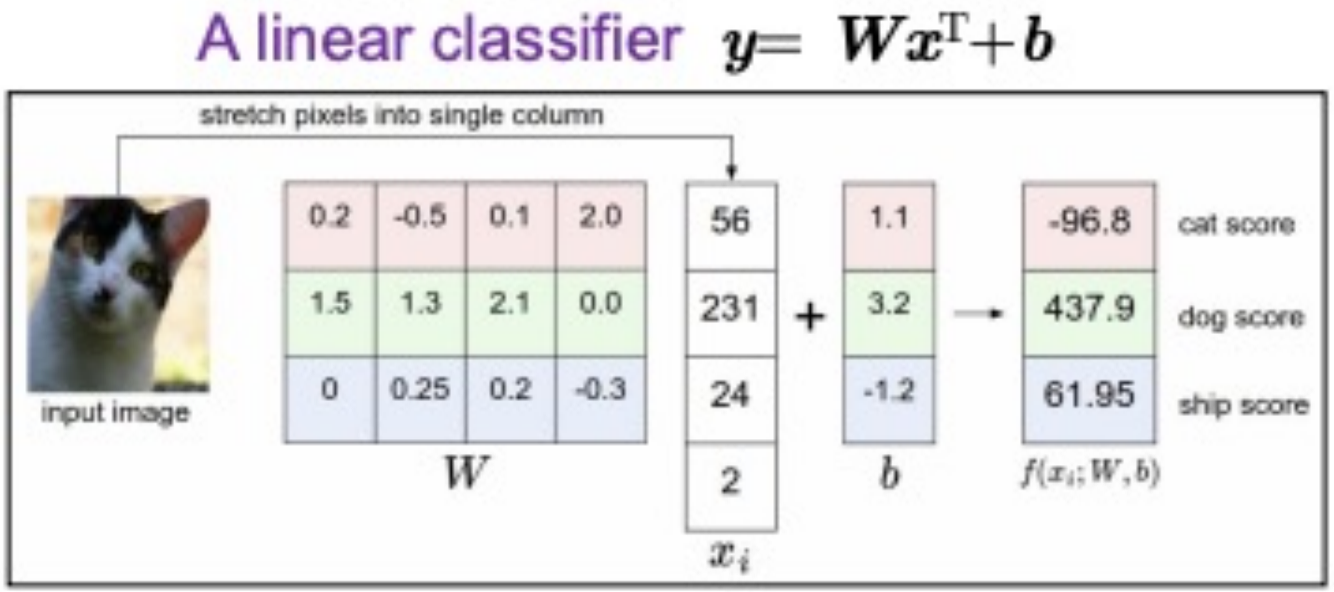
b → bias term (scalar)

- Very similar in the mathematical representation of a line

$$y = mx + c$$

→Hence the term ***linear classifier***

Linear Classifier (cntd.)



Example

- Suppose we have a dataset giving the living areas and prices of 47 houses from Stillwater, OK

Living area (ft^2)	# bedrooms	Price (1000$s)
1643	4	256
1356	3	202
1678	3	287
…	…	…
3000	4	400

Example

- Suppose we have a dataset giving the living areas and prices of 47 houses from Stillwater, OK

Living area (ft^2)	# bedrooms	Price (1000$s)
1643	4	256
1356	3	202
1678	3	287
…	…	…
3000	4	400

Features (X)

Targets (Y)

Example

- Suppose we have a dataset giving the living areas and prices of 47 houses from Stillwater, OK

Living area (ft^2)	# bedrooms	Price (1000$s)
1643	4	256
1356	3	202
1678	3	287
…	…	…
3000	4	400

Features (X)

Targets (Y)

Targets are continuous valued! => Task is *regression*

Linear Regression

- Goal: formulate a hypothesis function h(x) which will model the 2-d input feature (size, # bedrooms) and produce the expected target value (the house price in 1000$s).
- We can say that the hypothesis function could be a linear function of x:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

- Here, θ_i represent the ***parameters*** or ***weights*** of the linear model characterizing X→Y
- A more simpler model then will be $$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

Linear Regression (cntd.)

- **Learning:** Given this formulation, we will need to identify a way to find the values of θ.
- We will need to use the ***training*** data to learn these parameters. This process is called ***learning***
- **What do we need to achieve this?**

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

Linear Regression (cntd.)

- **Learning:** Given this formulation, we will need to identify a way to find the values of θ.
- We will need to use the ***training*** data to learn these parameters. This process is called ***learning***
- **What do we need to achieve this?**
- We will define a function that measures the quality of predictions for each value of θ.
- This is called the **cost function or objective function**

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

Linear Regression (cntd.)

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

- **Learning:** Given this formulation, we will need to identify a way to find the values of θ.
- We will need to use the ***training*** data to learn these parameters. This process is called ***learning***
- **What do we need to achieve this?**
- We will define a function that measures the quality of predictions for each value of θ.
- This is called the **cost function or objective function**

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2.$$

Linear Regression (cntd.)

- **Learning:** Given this formulation, we will need to identify a way to find the values of θ.
- We will need to use the ***training*** data to learn these parameters. This process is called ***learning***
- **What do we need to achieve this?**
- We will define a function that measures the quality of predictions for each value of θ.
- This is called the **cost function or objective function**

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2.$$

Ordinary least squares regression model

Linear Regression (cntd.)

- **Learning:** Given this formulation, we will need to identify a way to find the values of θ.
- We will need to use the ***training*** data to learn these parameters. This process is called ***learning***
- **What do we need to achieve this?**
- We will define a function that measures the quality of predictions for each value of θ.
- This is called the **cost function or objective function**

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

How to obtain parameter matrix using this objective function?

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2.$$

Ordinary least squares regression model

Learning

- Goal: To find a set of parameters θ that will minimize the cost function J(θ).
- Common approach: *gradient descent*

Learning

- Goal: To find a set of parameters θ that will minimize the cost function J(θ).
- Common approach: *gradient descent*
- What does it do?

Learning

- Goal: To find a set of parameters θ that will minimize the cost function J(θ).
- Common approach: *gradient descent*
- What does it do?
  - Start with an initial "guess" for θ
  - Update values of θ that will gradually move towards the "optimal solution"
  - What is the optimal solution?
  - The value of θ that minimizes the cost function
- How do we do it computationally?

Gradient Descent

- How do we do it computationally?

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

- α is the learning rate
  - Modulates how much of the change that we need to propagate at each instant
- Each update of θ will be a step in the *steepest decrease of the cost function J(*θ)
- How to Compute the derivative of the cost function J(θ)?

$$h_\theta(x) = \theta^T x$$

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} \left(\theta^T x^{(i)} - y^{(i)} \right)^2$$

$$\frac{d J(\theta)}{d\theta} = \frac{d}{d\theta} \left[\frac{1}{2} \sum_{i=1}^{m} \left(\theta^T x^{(i)} - y^{(i)} \right)^2 \right]$$

For convenience, $x^{(i)} \rightarrow x$, $y^{(i)} \rightarrow y$

and compute the summation over the derivatives.

$$\frac{dJ(\theta)}{d\theta} = \frac{d}{d\theta}\left[\frac{1}{2}(\theta^T x - y)^2\right]$$

$$= \frac{1}{2} \cdot \frac{d}{d\theta}\left[(\theta^T x) - y\right]^2$$

We know that $\frac{d}{dz}(z^n) = n \cdot z^{n-1}$

$$\Rightarrow \frac{d\,J(\theta)}{d\theta} = \frac{1}{2} \cdot \left[2 \cdot \frac{d}{d\theta}(\theta^T x) - \frac{d}{d\theta}(y) \right](\theta^T x - y)$$

we know that y is the true label

$\Rightarrow y$ is a constant

$$\therefore \frac{d\,J(\theta)}{d\theta} = \frac{2}{2}\left[x - 0 \right](\theta^T x - y)$$

$$\therefore \frac{d(J(\theta))}{d\theta} = (\theta^T x - y)\,x$$

Gradient Descent

- Hence each update is given by

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

- This is called the **LMS** update rule or the ***Least Mean Squares*** update rule.
  - Also known as the ***Widrow-Hoff*** learning rule.

Gradient Descent

- Has several properties:
  - Magnitude of update is proportional to the error $(y - h(x))$
    - What does this mean?

Gradient Descent

- Has several properties:
  - Magnitude of update is proportional to the error (y – h(x))
    - What does this mean?
    - If we have a very good prediction i.e. h(x)$\approx y$, then the update is very small.
    - Conversely, if the prediction is very far off i.e. h(x)$\gg y$ or h(x)$\ll y$ then the update will be large.
- For learning over the complete training set, we iteratively update the parameters as below

Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}

Gradient Descent

- If we use all the examples in the training set ***at once***:
  - Batch gradient descent
- What if we update at every single data point?
  - Stochastic or incremental gradient descent

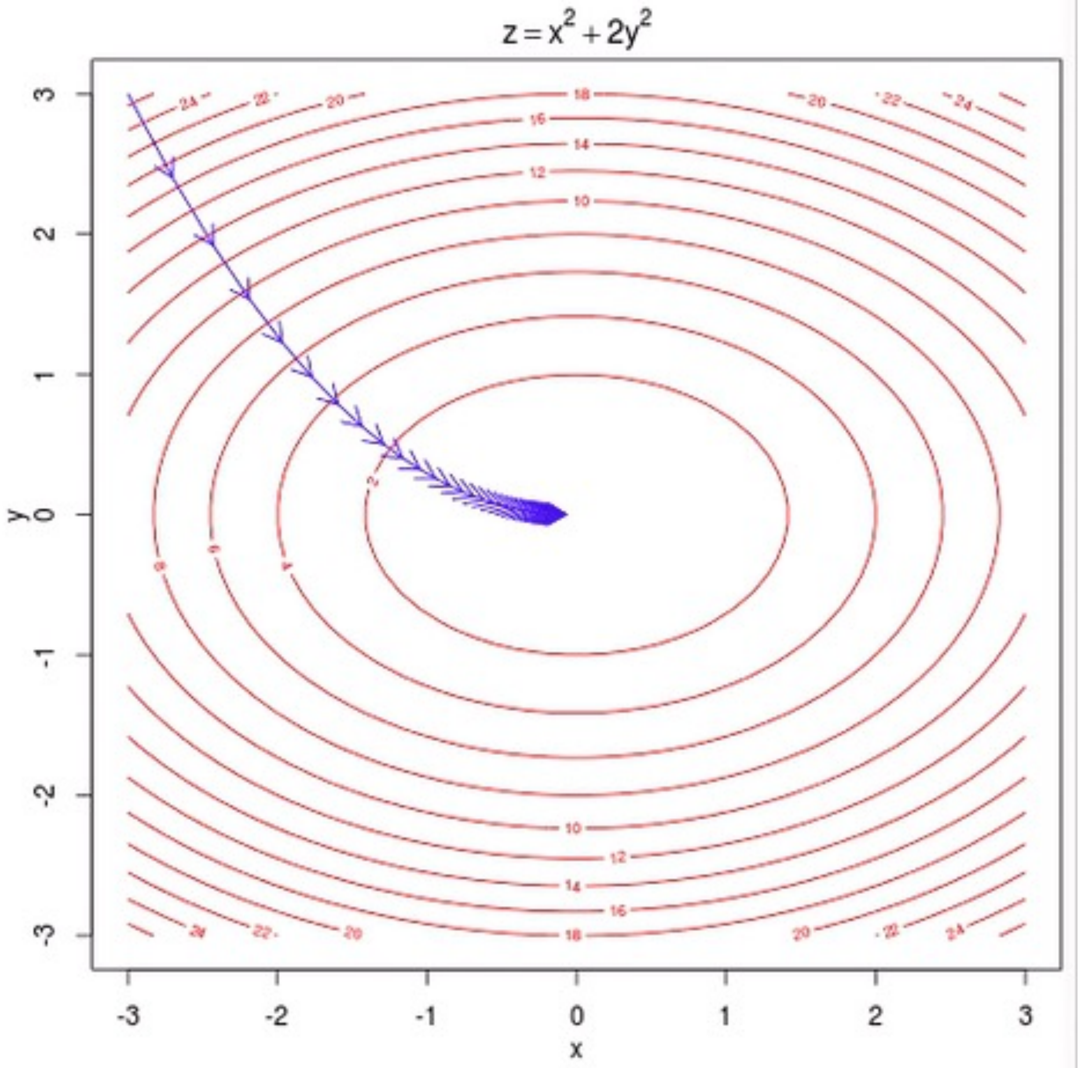
Loop {

for i=1 to m, {

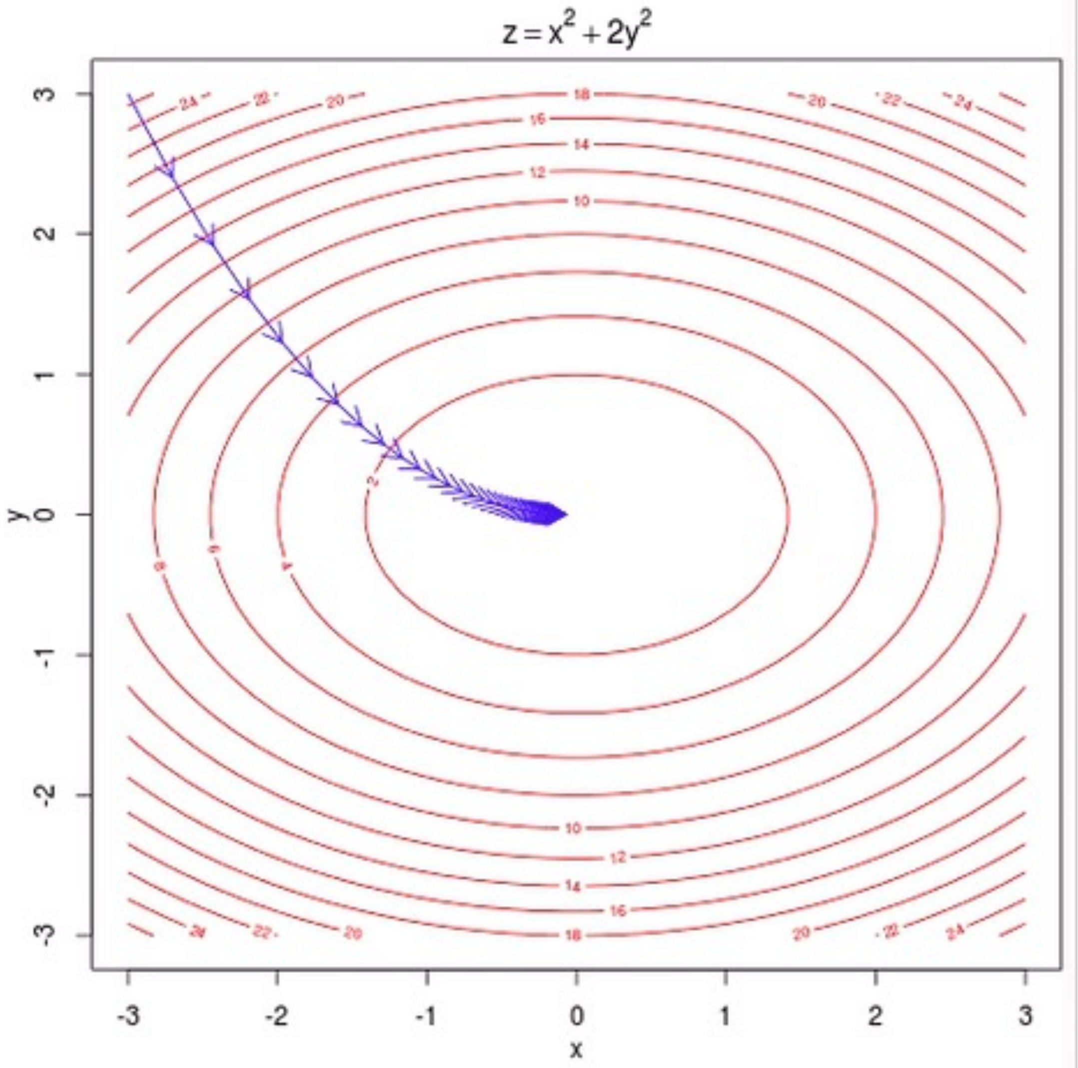
$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}

}



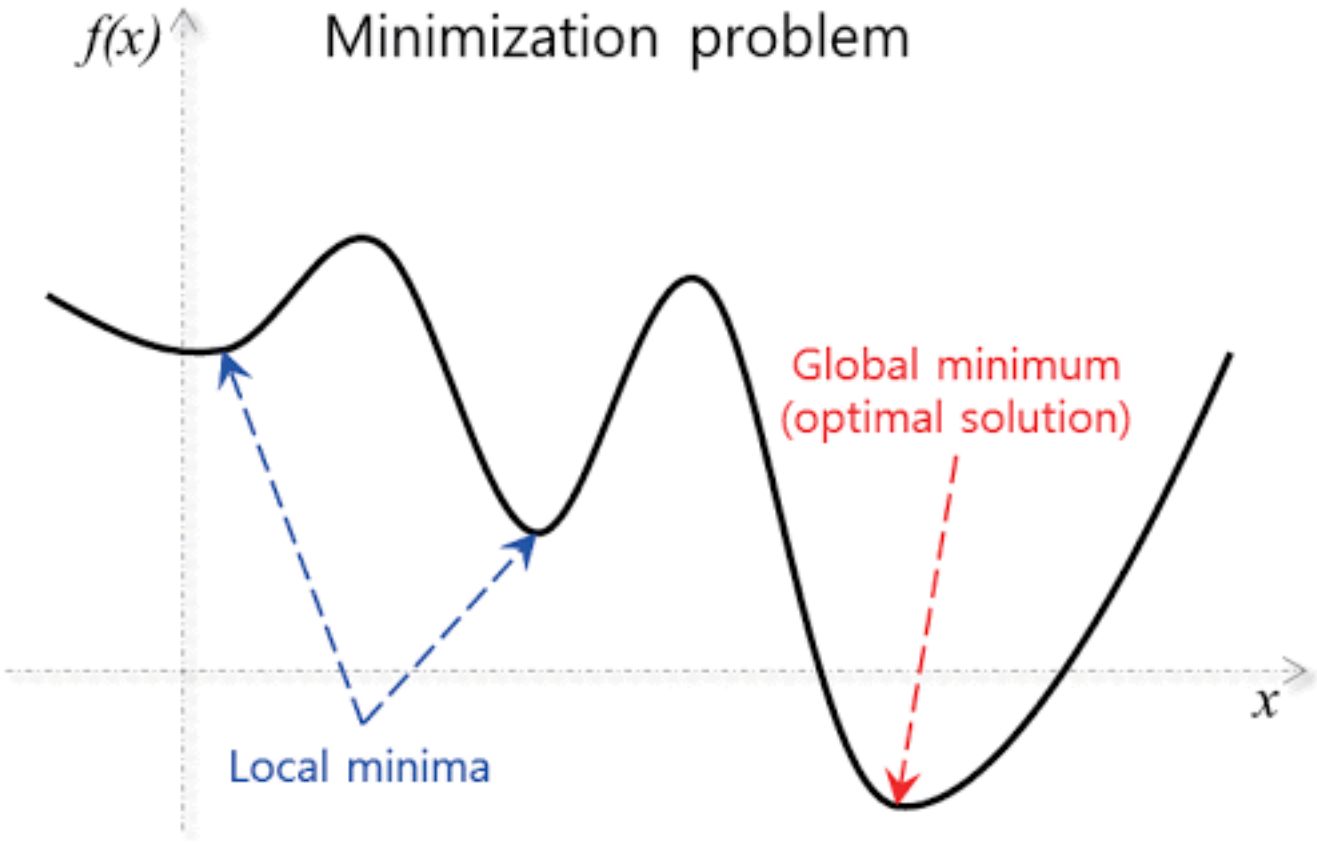
- Gradient Descent in 2D. Images Credit: http://vis.supstat.com/2013/03/gradient-descent-algorithm-with-r/



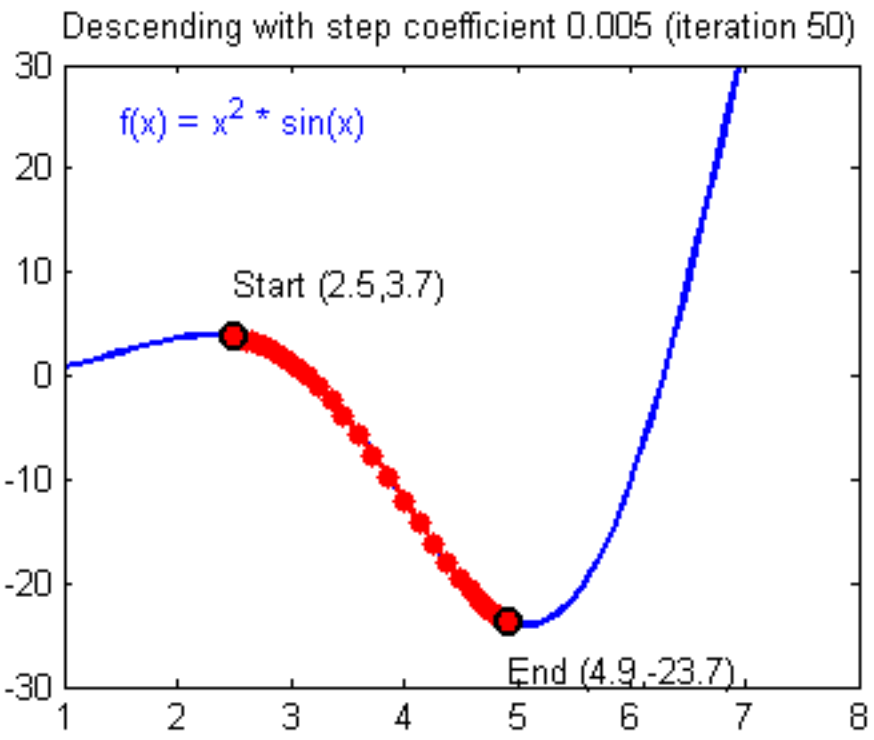
Will we always get an optimal solution?

- Gradient Descent in 2D. Images Credit: http://vis.supstat.com/2013/03/gradient-descent-algorithm-with-r/

Not really…

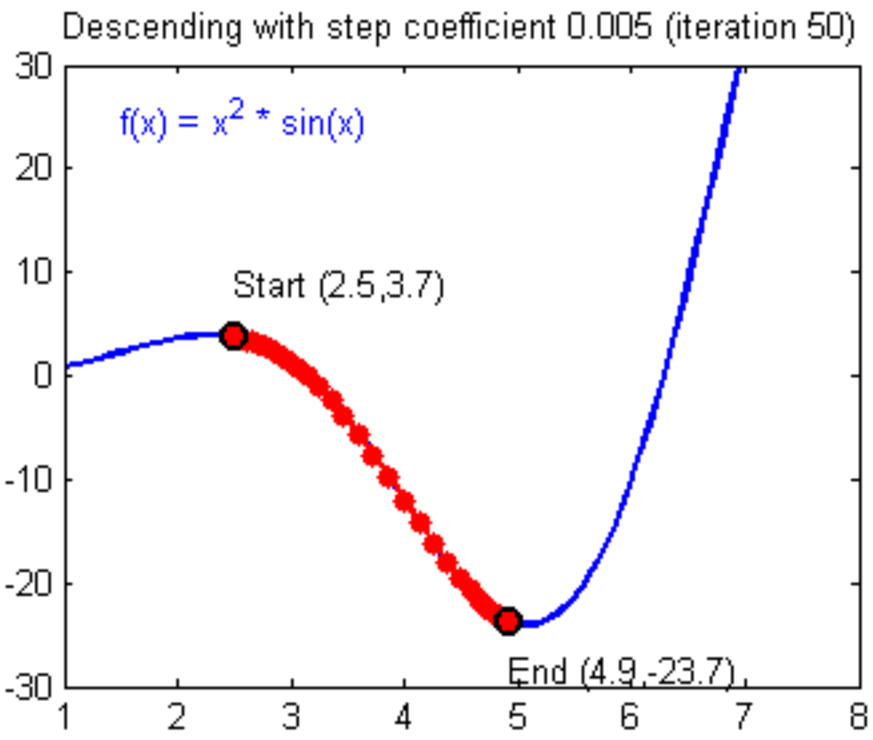

f(x)
Minimization problem
Global minimum
(optimal solution)
Local minima
x

Is the learning rate that important?

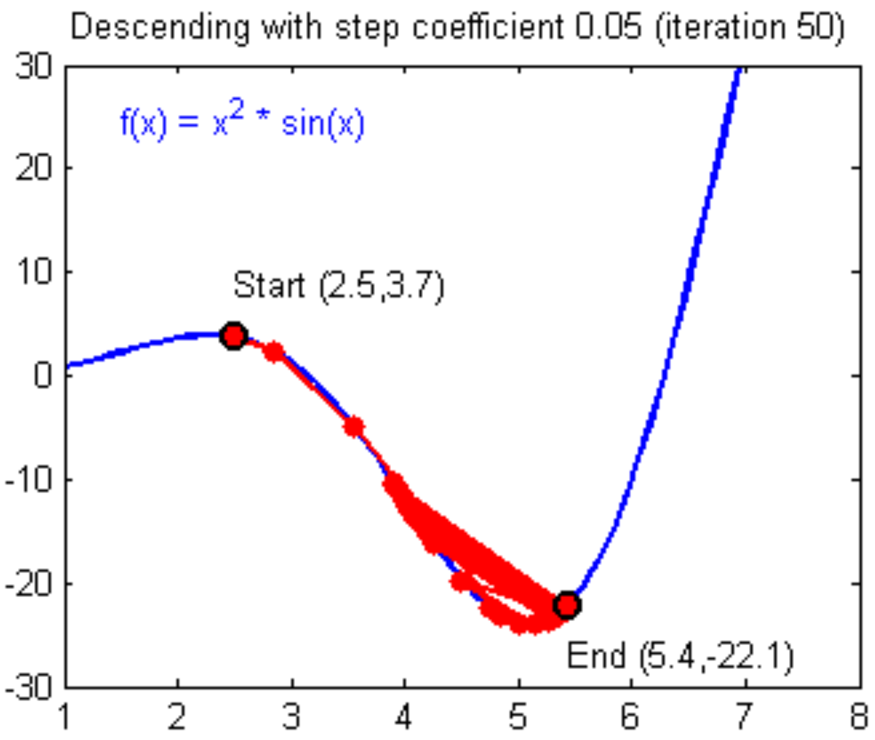


Smaller Learning Rate

Is the learning rate that important?



Smaller Learning Rate



Larger Learning Rate

Is the learning rate that important?

Is there a non-iterative solution?

Is there a non-iterative solution?

- Yes! We can directly find the solution using “Normal Equations”.
- It is an analytical approach used for optimization
- Can be done as follows:
  - Set the partial derivatives of the cost function J(θ) to zero
    - Remember, vectorized version of J(θ) = $\frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y})$
    - Hence, taking the Partial Derivative gives us

$$\nabla_\theta J(\theta) = X^T X\theta - X^T\vec{y}$$

Is there a non-iterative solution?

- Yes! We can directly find the solution using “Normal Equations”.
- It is an analytical approach used for optimization
- Can be done as follows:
  - Set the partial derivatives of the cost function $J(\theta)$ to zero
  - Then you can estimate the parameters as follows:

$$\nabla_\theta J(\theta) = X^T X\theta = X^T \vec{y}$$

$$\Theta = (X^T X)^{-1} X^T y$$

  - Where X is the input feature vector
  - y is the expected target value

Linear Regression with Basis Functions

Basis Functions

- Sometimes, you will need to have some fixed pre-processing of the input data.
  - Also called feature extraction
- Linearity is often a good assumption when many inputs influence the output
  - Natural examples include $F = ma$

Basis Functions

- Sometimes, you will need to have some fixed pre-processing of the input data.
  - Also called feature extraction
- Linearity is often a good assumption when many inputs influence the output
  - Natural examples include $F = ma$
- But in general, it is rather unlikely that a true function is linear

Basis Functions

- Sometimes, you will need to have some fixed pre-processing of the input data.
  - Also called feature extraction
- Linearity is often a good assumption when many inputs influence the output
  - Natural examples include $F = ma$
- But in general, it is rather unlikely that a true function is linear

How do we handle non-linear relationships?

Basis Functions

- Sometimes, you will need to have some fixed pre-processing of the input data.
  - Also called feature extraction
- Linearity is often a good assumption when many inputs influence the output
  - Natural examples include $F = ma$
- But in general, it is rather unlikely that a true function is linear

How do we handle non-linear relationships?

- Simple! In addition to the original features, add more features that are deterministic functions of the original features

Basis Functions

- If the original variables comprise the vector x, then the features can be expressed in terms of basis functions $\{\phi_j(x)\}$
- By using nonlinear basis functions, we allow the hypothesis function h(x,w) to be a nonlinear function of the input vector x
  - They are linear functions of parameters, yet are nonlinear with respect to the input variables
- You can have multiple basis functions to model different non-linearities

$$y(\mathrm{x},\mathrm{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathrm{x})$$

Basis Functions

- If the original variables comprise the vector x, then the features can be expressed in terms of basis functions $\{\phi_j(x)\}$
- By using nonlinear basis functions, we allow the hypothesis function h(x,w) to be a nonlinear function of the input vector x
  - They are linear functions of parameters, yet are nonlinear with respect to the input variables
- You can have multiple basis functions to model different non-linearities

$$y(\mathrm{x},\mathrm{w}) = w_0 + \sum_{j=1}^{M-1} w_j \underbrace{\phi_j(\mathrm{x})}_{\text{Basis functions}}$$

Basis Functions

- More generally,

$$y(\mathbf{x},\mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

where $w=(w_0,w_1,..,w_{M-1})$ and $\Phi=(\phi_0,\phi_1,..,\phi_{M-1})^T$

- We now need M weights for basis functions instead of D weights for features

Types of Basis Functions

- Linear Basis function:

$$y(x,\mathrm{w}) = \sum_{j=0}^{M-1} w_j \phi_j(x) = \mathrm{w}^T \phi(x)$$

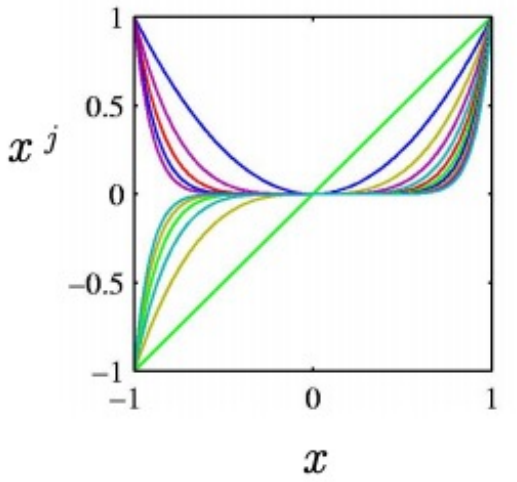
- Polynomial Basis Function:
  - Model with M-1 degree polynomial $\boxed{\phi_j(x) = x^j}$
  - Works really well for 1-d feature vector i.e. only modelling one variable *x*

Types of Basis Functions

- Linear Basis function:

$$y(x,\mathrm{w}) = \sum_{j=0}^{M-1} w_j \phi_j(x) = \mathrm{w}^T \phi(x)$$

- Polynomial Basis Function:
  - Model with M-1 degree polynomial $\boxed{\phi_j(x) = x^j}$
  - Works really well for 1-d feature vector i.e. only modelling one variable *x*

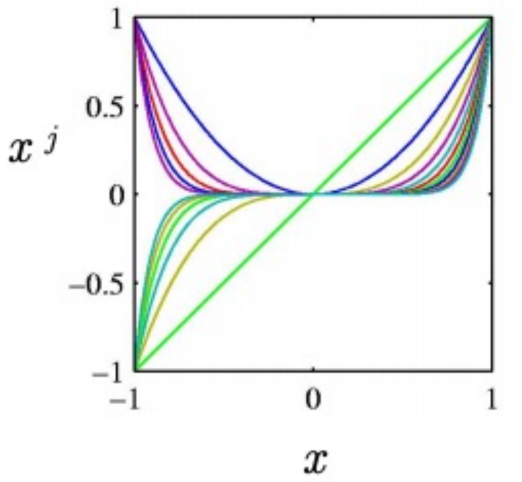


Types of Basis Functions

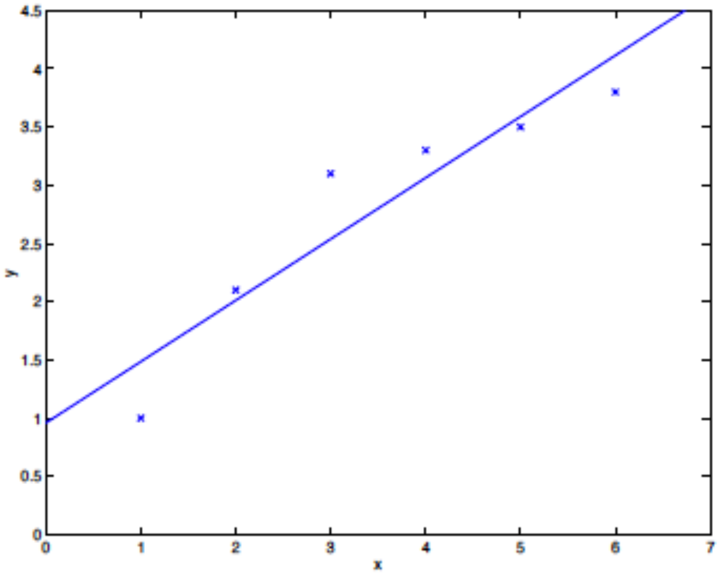
- Linear Basis function:

$$y(x,\mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(x) = \mathbf{w}^T \phi(x)$$

- Polynomial Basis Function:
  - Model with M-1 degree polynomial $\boxed{\phi_j(x) = x^j}$
  - Works really well for 1-d feature vector i.e. only modelling one variable *x*
- Disadvantage
  - Global:
    - changes in one region of input space affects others
  - Difficult to formulate
    - Number of polynomials increases exponentially with M
  - Can divide input space into regions
    - use different polynomials in each region
    - equivalent to spline functions

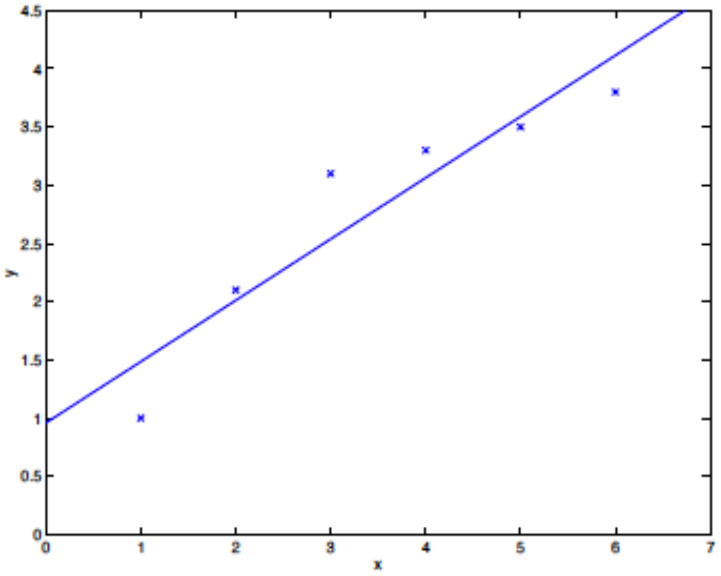


Example

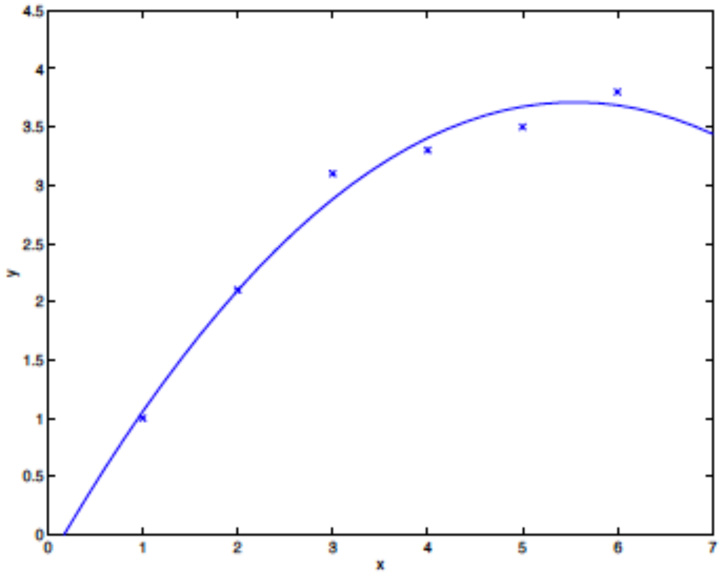


Linear Basis Function

Example

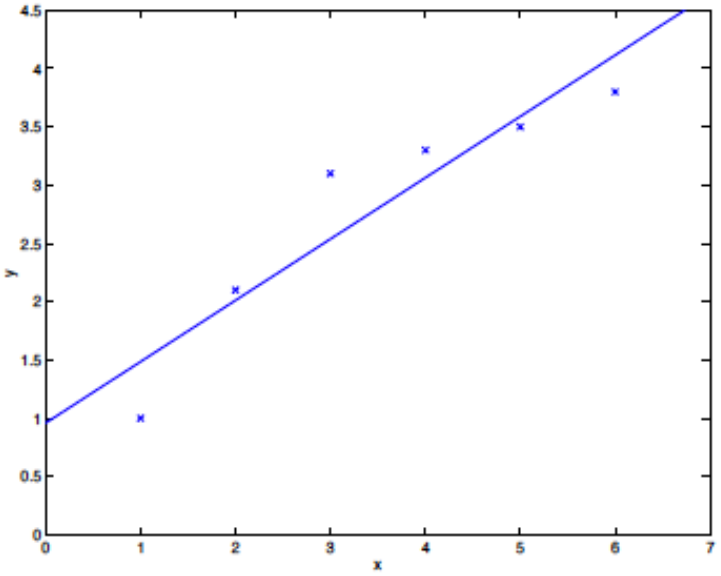


Linear Basis Function

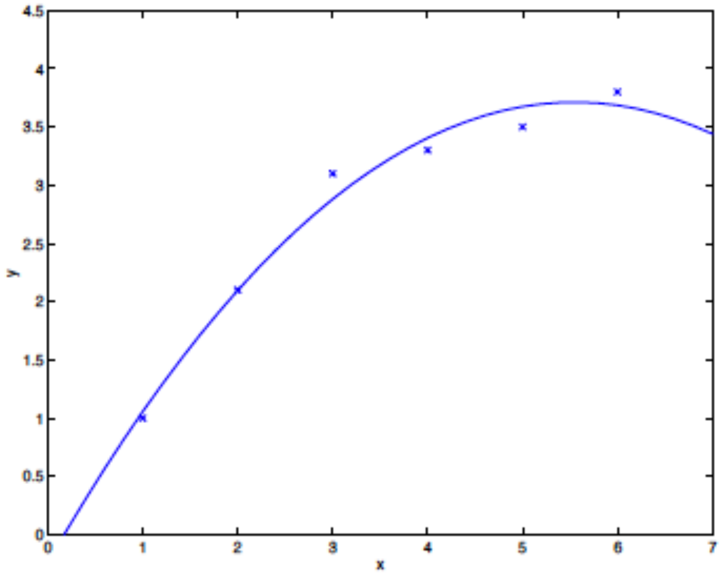


2nd order basis Function
i.e. add x^2 feature

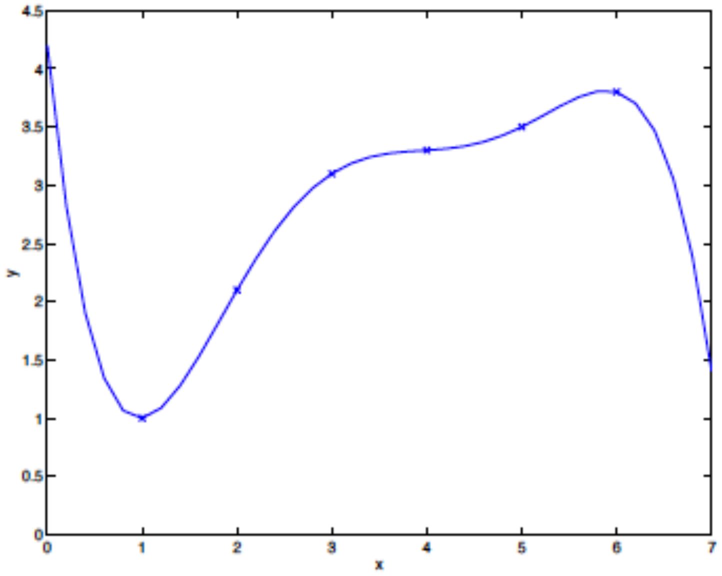
Example



Linear Basis Function



2^{nd} order basis Function
i.e. add x^2 feature



5^{th} order basis Function
i.e. add up to x^5 feature

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

Other Basis Functions

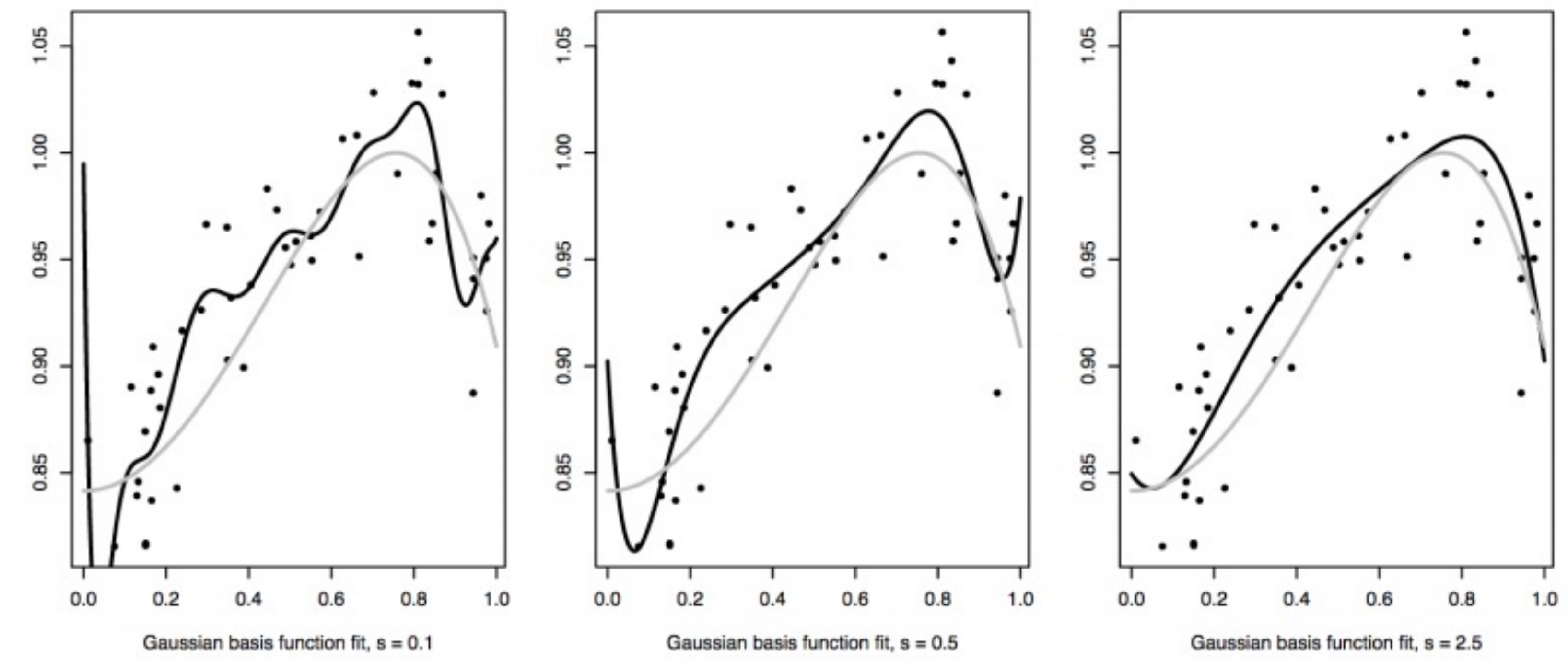
- Gaussian Radial Basis function:
  - μ_j govern the locations of the basis functions
    - Can be an arbitrary set of points within the range of the data
      - Can choose some representative data points
  - σ governs the spatial scale
    - Could be chosen from the data set e.g., average variance

$$\phi_j(x) = \exp\left(\frac{(x-\mu_j)^2}{2\sigma^2}\right)$$

Other Basis Functions

- Gaussian Radial Basis function:
  - μ_j govern the locations of the basis functions
  - σ governs the spatial scale
    - Could be chosen from the data set e.g., average variance

$$\phi_j(x) = \exp\left(\frac{(x-\mu_j)^2}{2\sigma^2}\right)$$

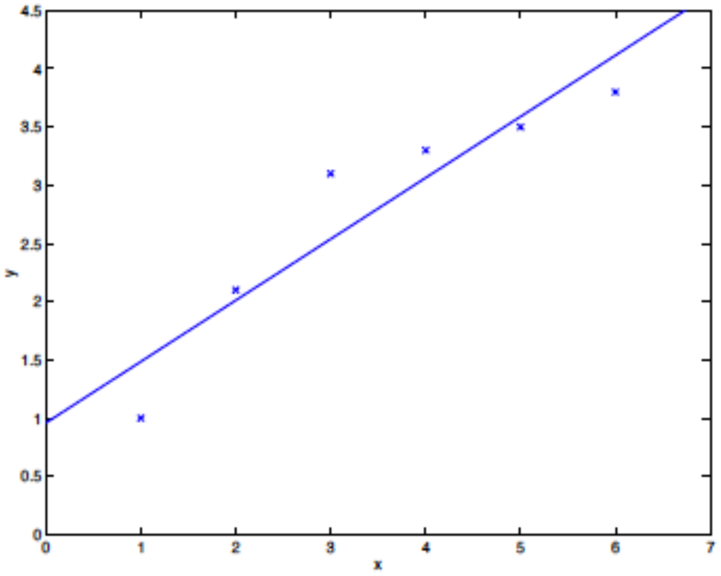


Other Basis Functions

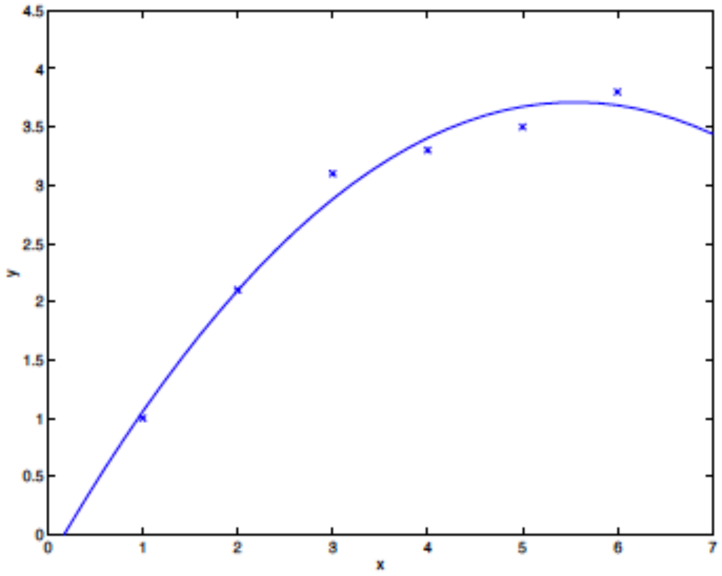
- Sigmoid: $\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$
- Fourier
  - Expansion in sinusoidal functions
- Signal Processing
  - Also called *wavelets*
    - Functions grounded in time and frequency
- Linear
  - $\phi(x) = x$

Does more features always mean better?

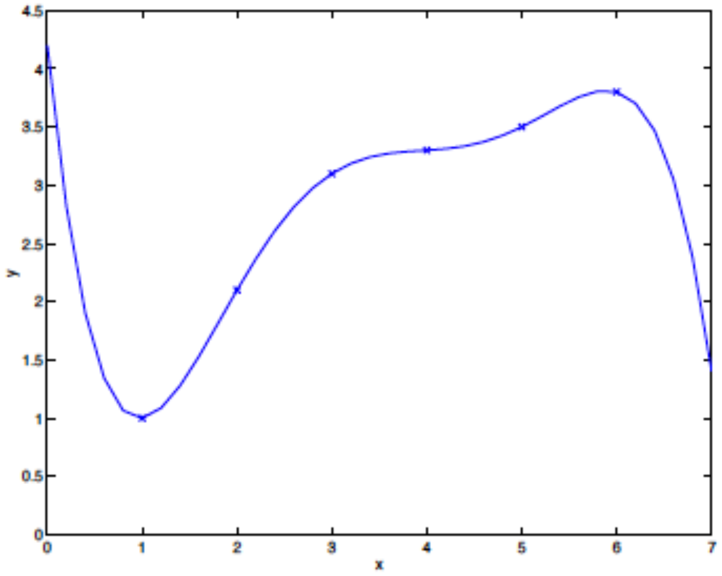
Does more features always mean better?



Linear Basis Function



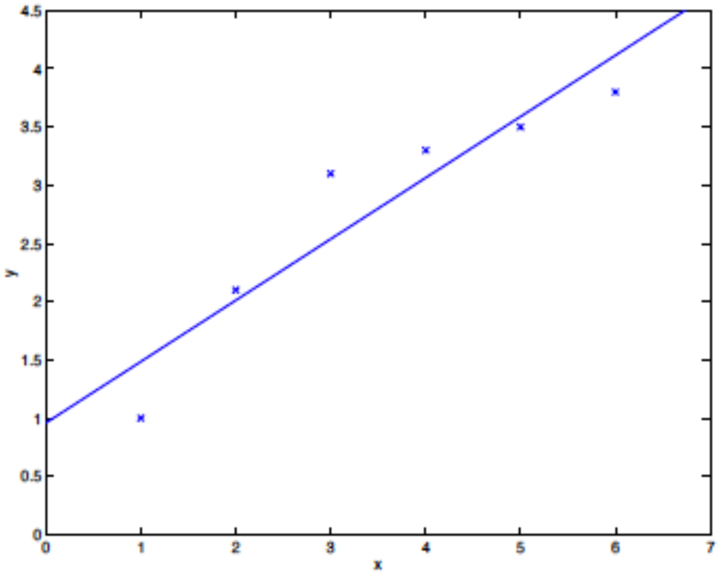
2^{nd} order basis Function
i.e. add x^2 feature



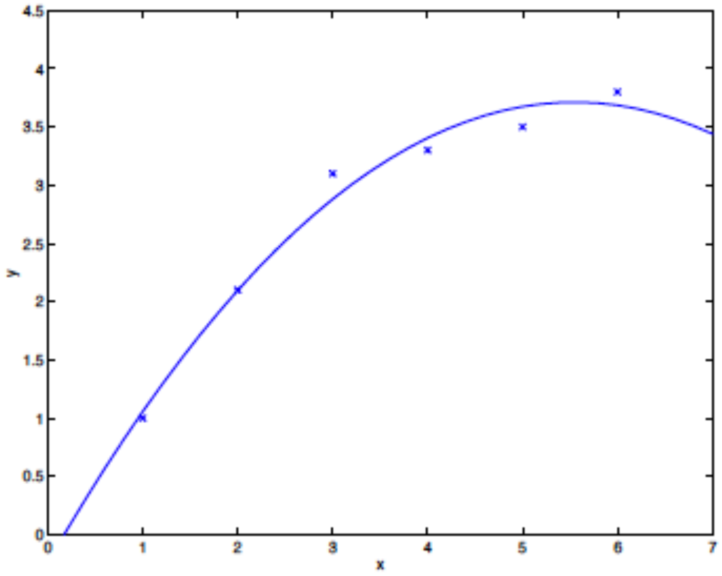
5^{th} order basis Function
i.e. add up to x^5 feature

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

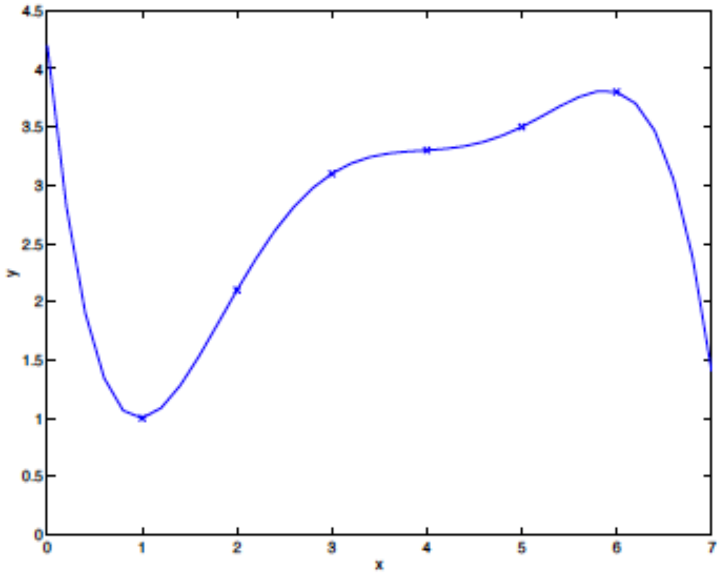
Does more features always mean better?



Linear Basis Function



2nd order basis Function
i.e. add x^2 feature

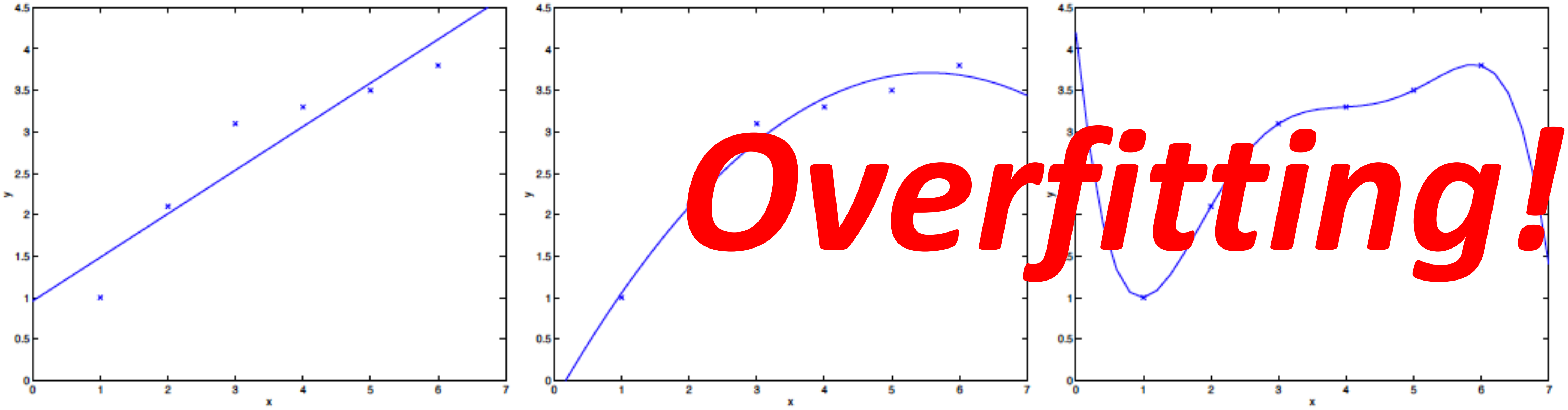


5th order basis Function
i.e. add up to x^5 feature

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

NO!

Does more features always mean better?



Overfitting!

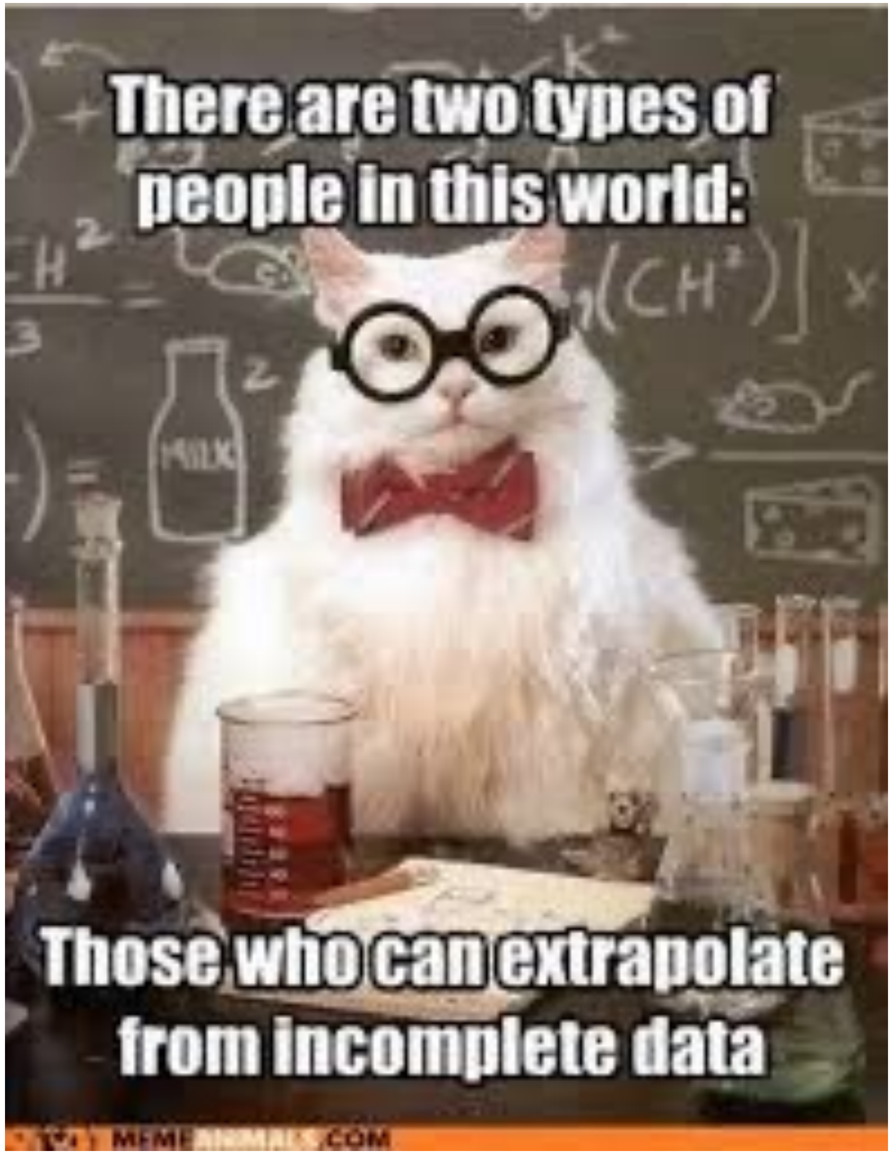
Linear Basis Function

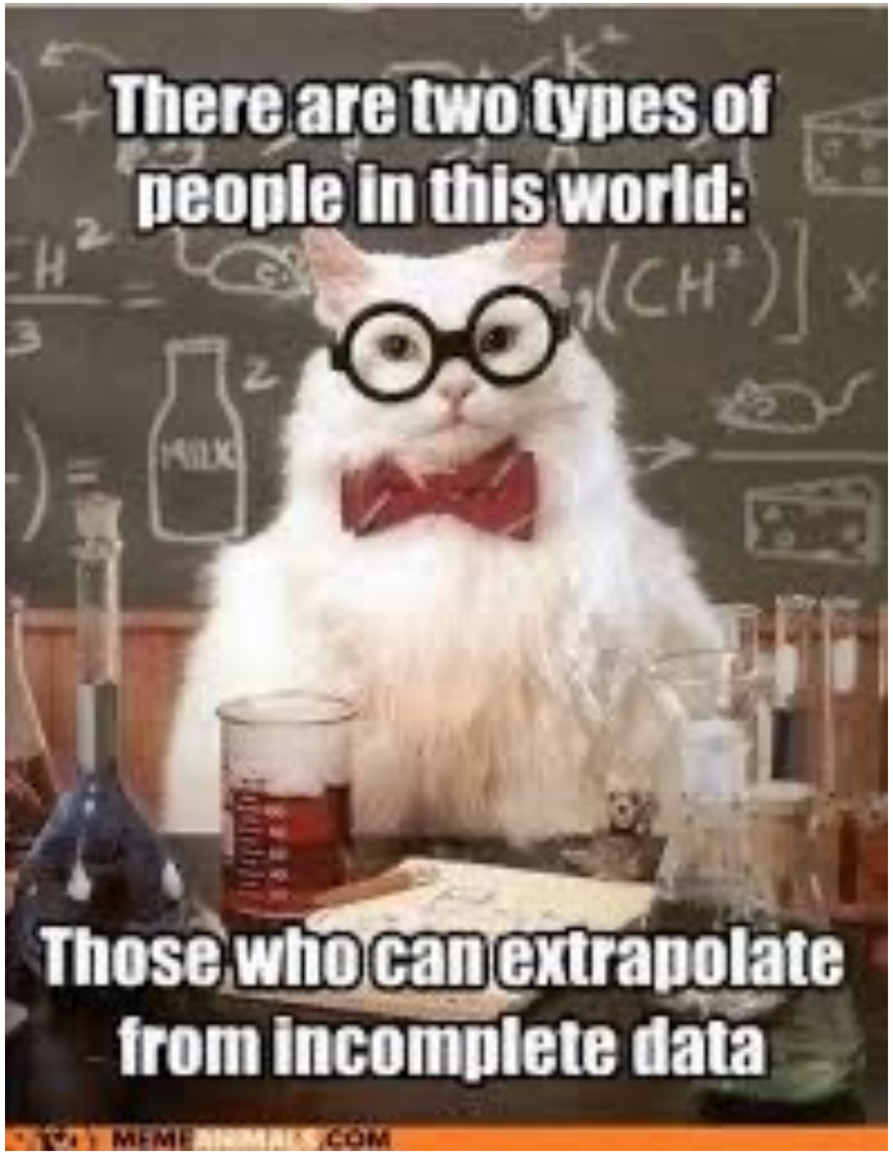
2^{nd} order basis Function
i.e. add x^2 feature

5^{th} order basis Function
i.e. add up to x^5 feature

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

NO!


There are two types of people in this world:
Those who can extrapolate from incomplete data



You don't want your model to be other kind!

☺