Data Mining: Artificial Neural Network

Lecture Notes for Chapter 3

Data Mining

https://ml-graph.github.io/winter-2025/

Yu Wang, Ph.D.

yuwang@uoregon.edu

Assistant Professor

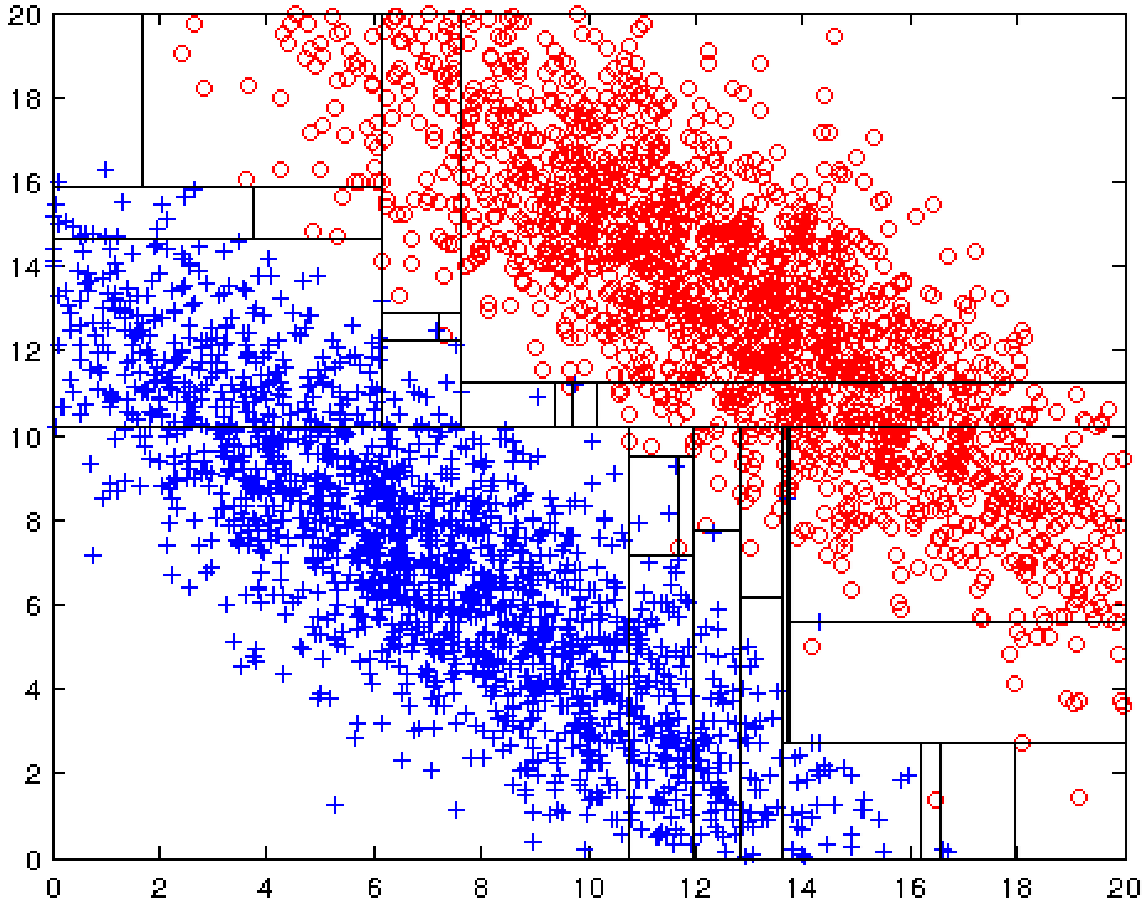
Computer Science

University of Oregon

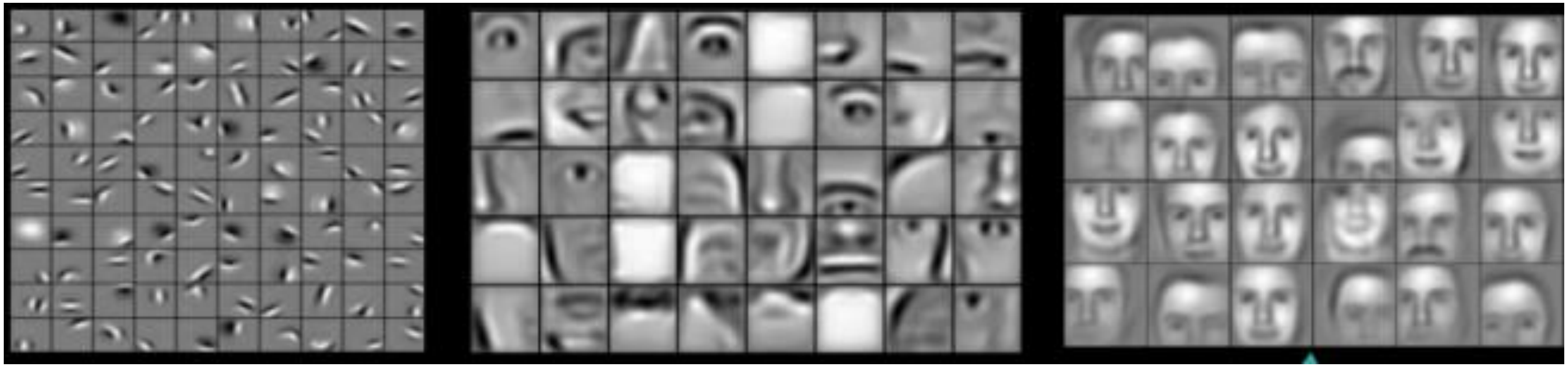
CS 453/553 – Winter 2025

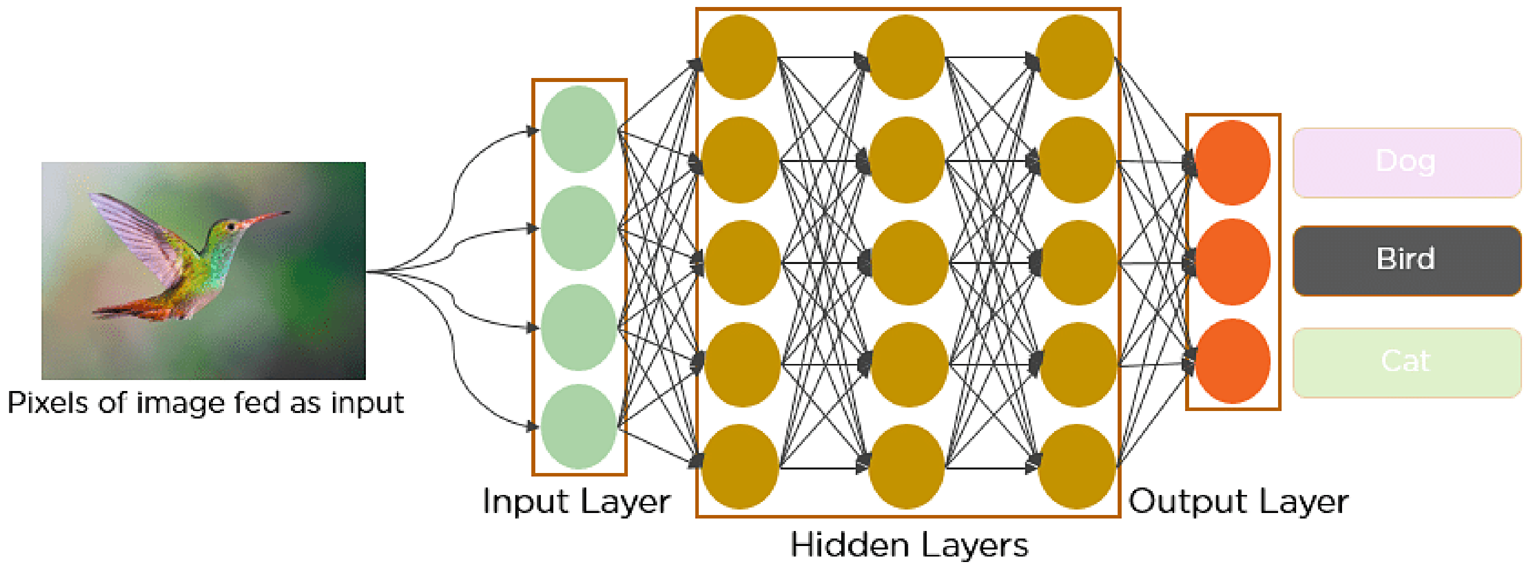
Course Lecture is very heavily based on
"Introduction to Data Mining"
by Tan, Steinbach, Karpatne, Kumar

Limitations of Decision-Tree and many other model

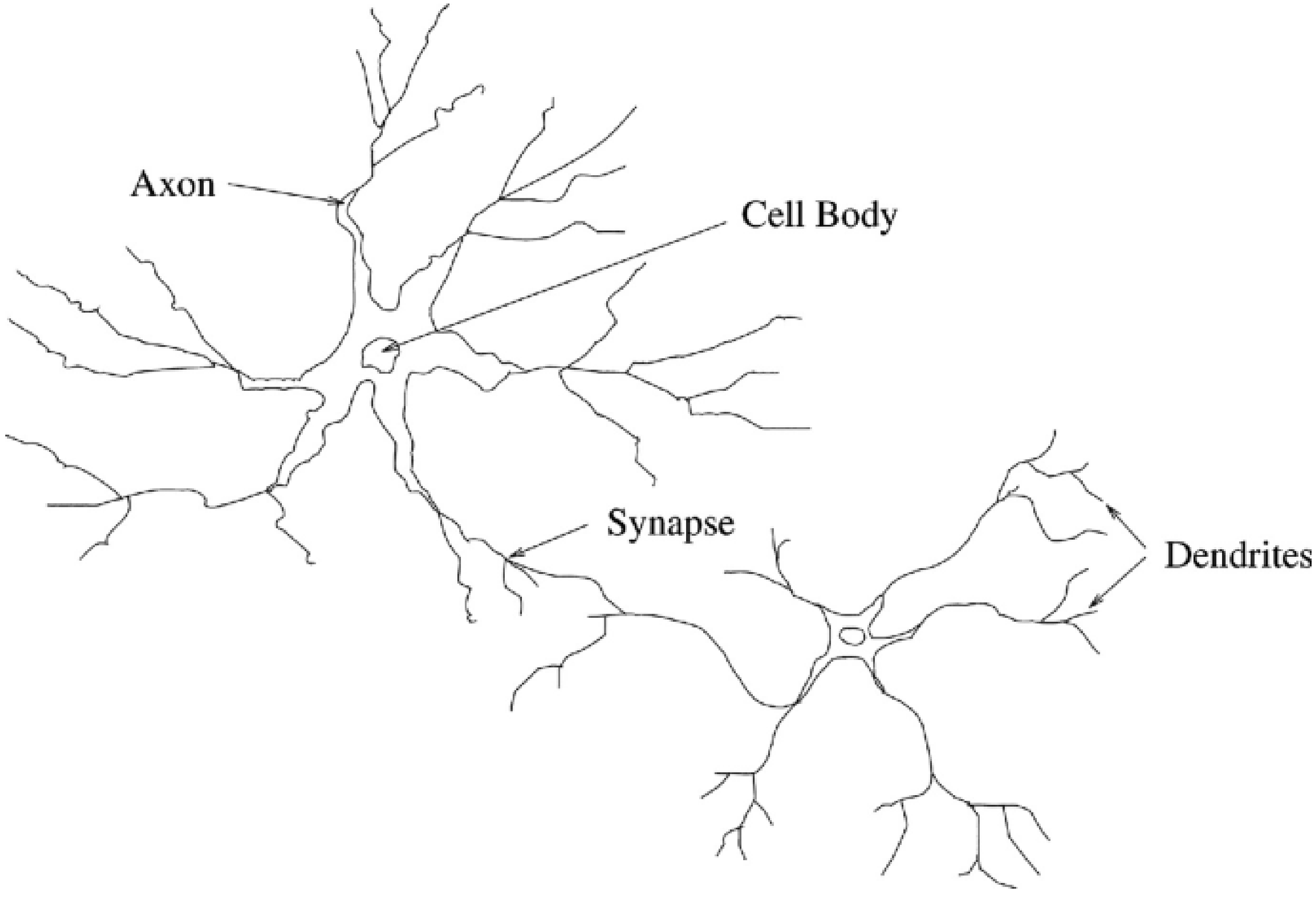


Real-world Intuition





Basic Architecture of Perceptron



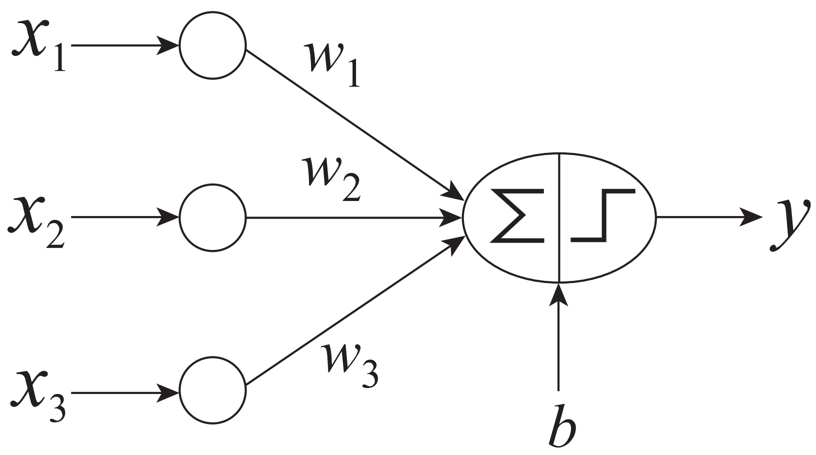
Artificial Neural Networks (ANN)

- Basic Idea: A complex non-linear function can be learned as a composition of simple processing units

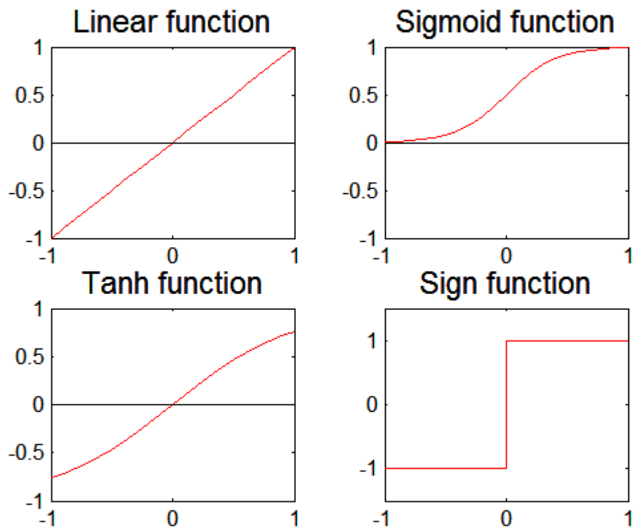
- ANN is a collection of simple processing units (nodes) that are connected by directed links (edges)
  - Every node receives signals from incoming edges, performs computations, and transmits signals to outgoing edges
  - Analogous to *human brain* where nodes are neurons and signals are electrical impulses
  - Weight of an edge determines the strength of connection between the nodes

- Simplest ANN: Perceptron (single neuron)

Basic Architecture of Perceptron



$$y = \sigma(w^{\mathrm{T}}x + b)$$



What happens if there is no nonlinear activation?

Linear Regression

- Data - $\left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$

$$f(x) = xw + b$$

- Regression – Find f that minimizes our uncertainty about y given x

$$y = f(x) + n$$

- Minimizing Mean Squared Error = Minimizing Negative Log-Likelihood

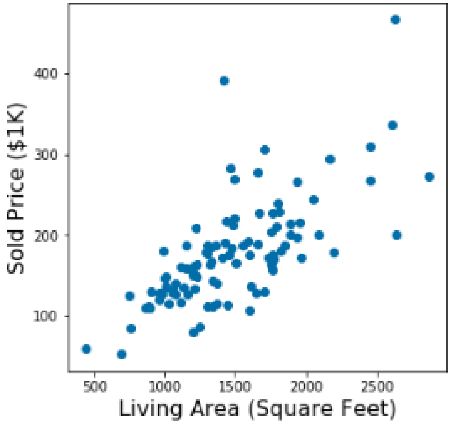
$$\underset{f}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - f\left(x^{(i)}\right)\right)^2$$

Linear Regression

$$\underset{w}{\operatorname{argmin}} \boxed{\frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - w^T\hat{x}^{(i)}\right)^2} = \underset{w}{\operatorname{argmin}} \boxed{\frac{1}{N}\|y - Xw\|^2}$$

Loss/Cost Function

$$\underset{f}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - f\left(x^{(i)}\right)\right)^2$$



Where

- $y = \left[y^{(1)}, \ldots, y^{(N)}\right]^T \in \mathbb{R}^{N\times 1}$ and
- $X = \left[\hat{x}^{(1)}, \ldots, \hat{x}^{(N)}\right]^T \in \mathbb{R}^{N\times(d+1)}$ (here $d = 1$)
- $w = [w_0, w_1, \ldots, w_d]^{\mathrm{T}} \in \mathbb{R}^{d+1}$

Linear Regression

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

Compute the minimum value? How to do it in Math?

Find points where gradient = 0

Linear Regression

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

$$J(W) = \frac{1}{N}(Y - XW)^T(Y - XW)$$

https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

$$J(W) = \frac{1}{N}\left(Y^TY - 2Y^TXW + W^TX^TXW\right)$$

$$\nabla J(W) = \frac{\partial}{\partial W}\left[\frac{1}{N}(Y^TY - 2Y^TXW + W^TX^TXW)\right]$$

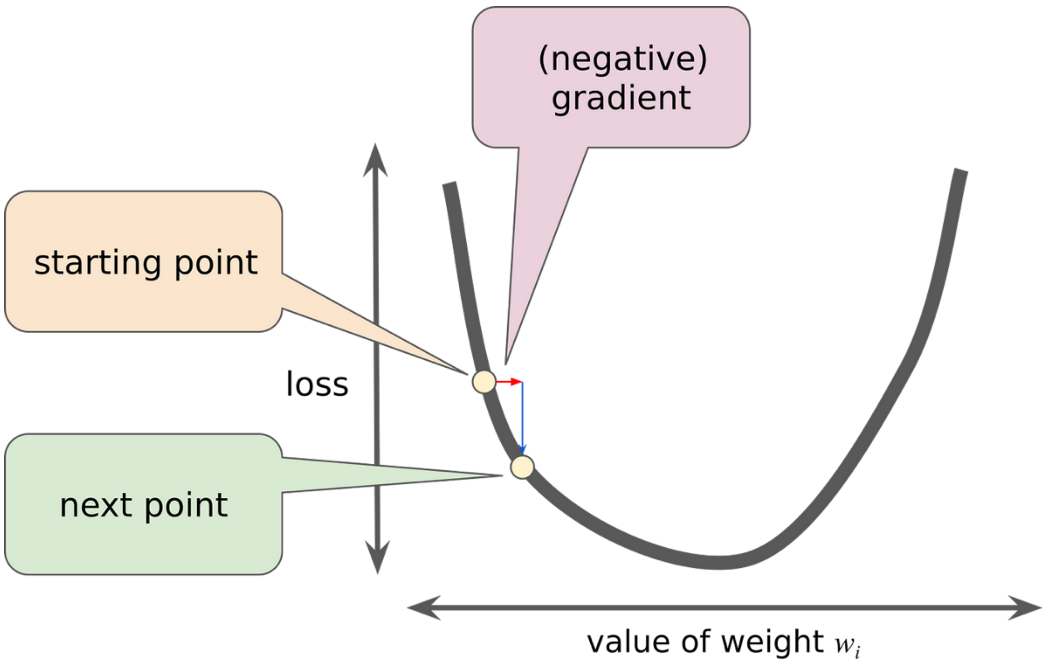
$$X^TXW = X^TY$$

$$W^* = (X^TX)^{-1}X^TY$$

$$\nabla J(W) = -\frac{2}{N}X^TY + \frac{2}{N}X^TXW$$

Linear Regression

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



If the gradient of a function is non-zero at a point, the direction of the gradient is the direction in which the function increases most quickly

Linear Regression

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

$$\nabla J(w) = \frac{\partial J(w)}{\partial w}$$

$$J(w) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - w^T\hat{x}^{(i)})^2$$

$$\frac{\partial J(w)}{\partial w} = \frac{1}{N}\sum_{i=1}^{N} 2(y^{(i)} - w^T\hat{x}^{(i)})(-\hat{x}^{(i)})$$

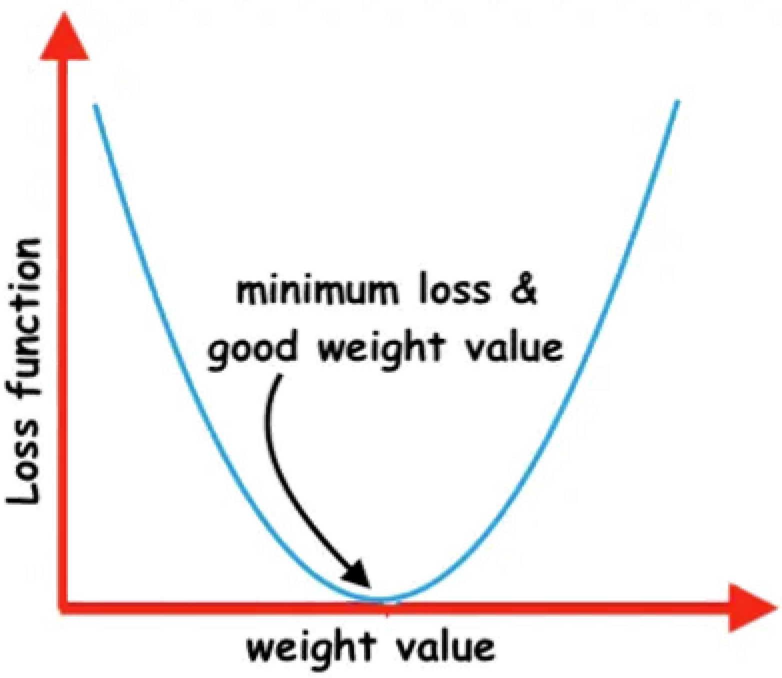
$$= -\frac{2}{N}\sum_{i=1}^{N}(y^{(i)} - w^T\hat{x}^{(i)})\hat{x}^{(i)}$$

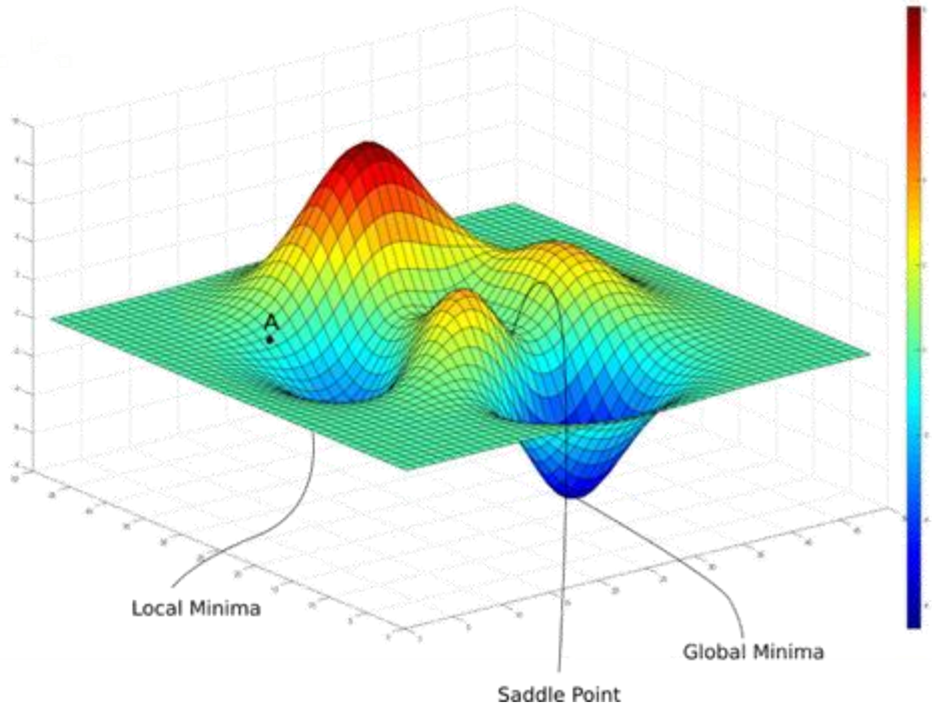
$$\nabla J(w) = -\frac{2}{N}\sum_{i=1}^{N}(y^{(i)} - w^T\hat{x}^{(i)})\hat{x}^{(i)}$$

$$w_0 := w_0 + \frac{2\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - (w_0 + w_1 x_1^{(i)}))$$

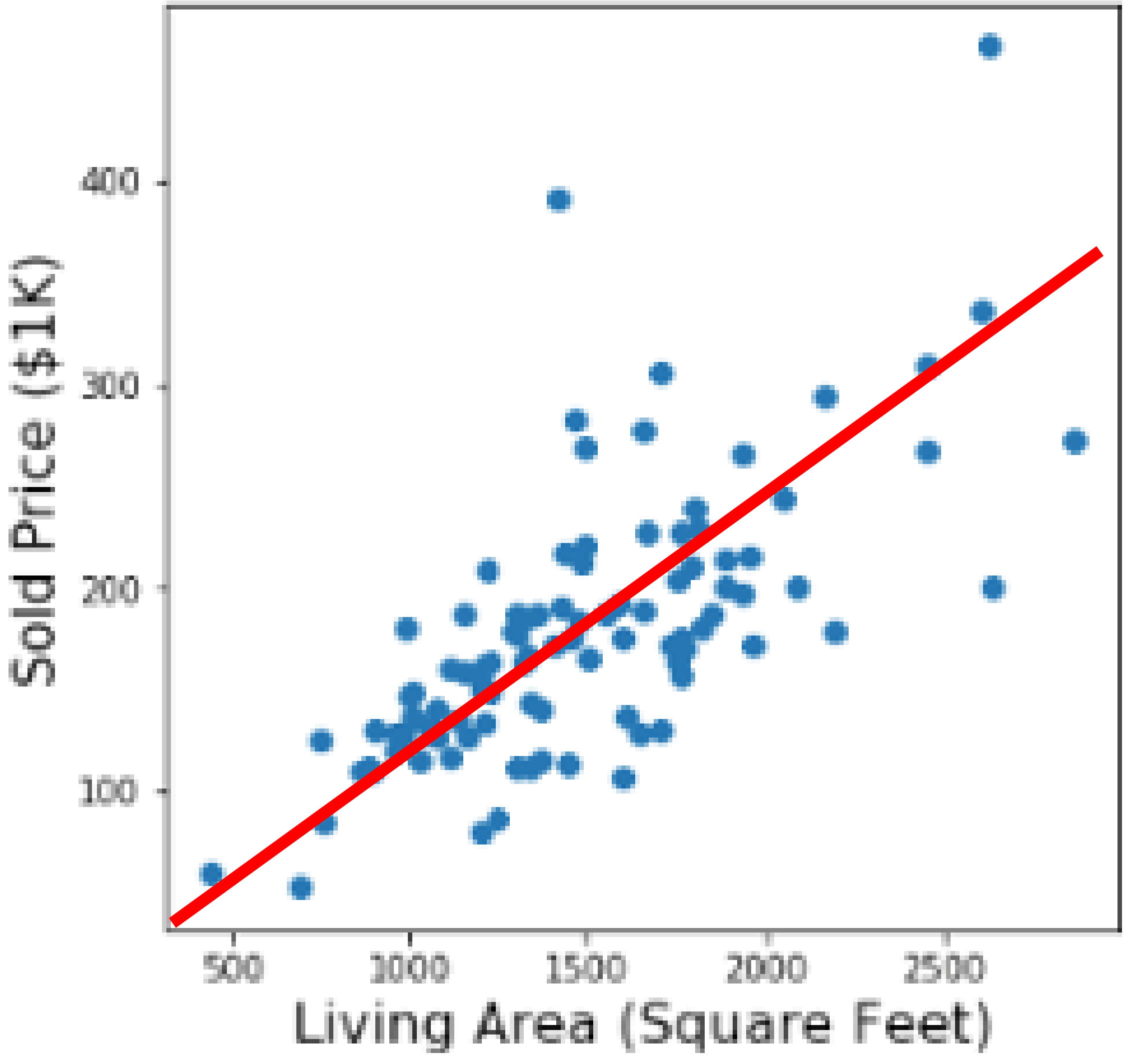
$$w_1 := w_1 + \frac{2\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - (w_0 + w_1 x_1^{(i)}))x_1^{(i)}$$

Local Minimum vs Global Minimum





Linear Regression

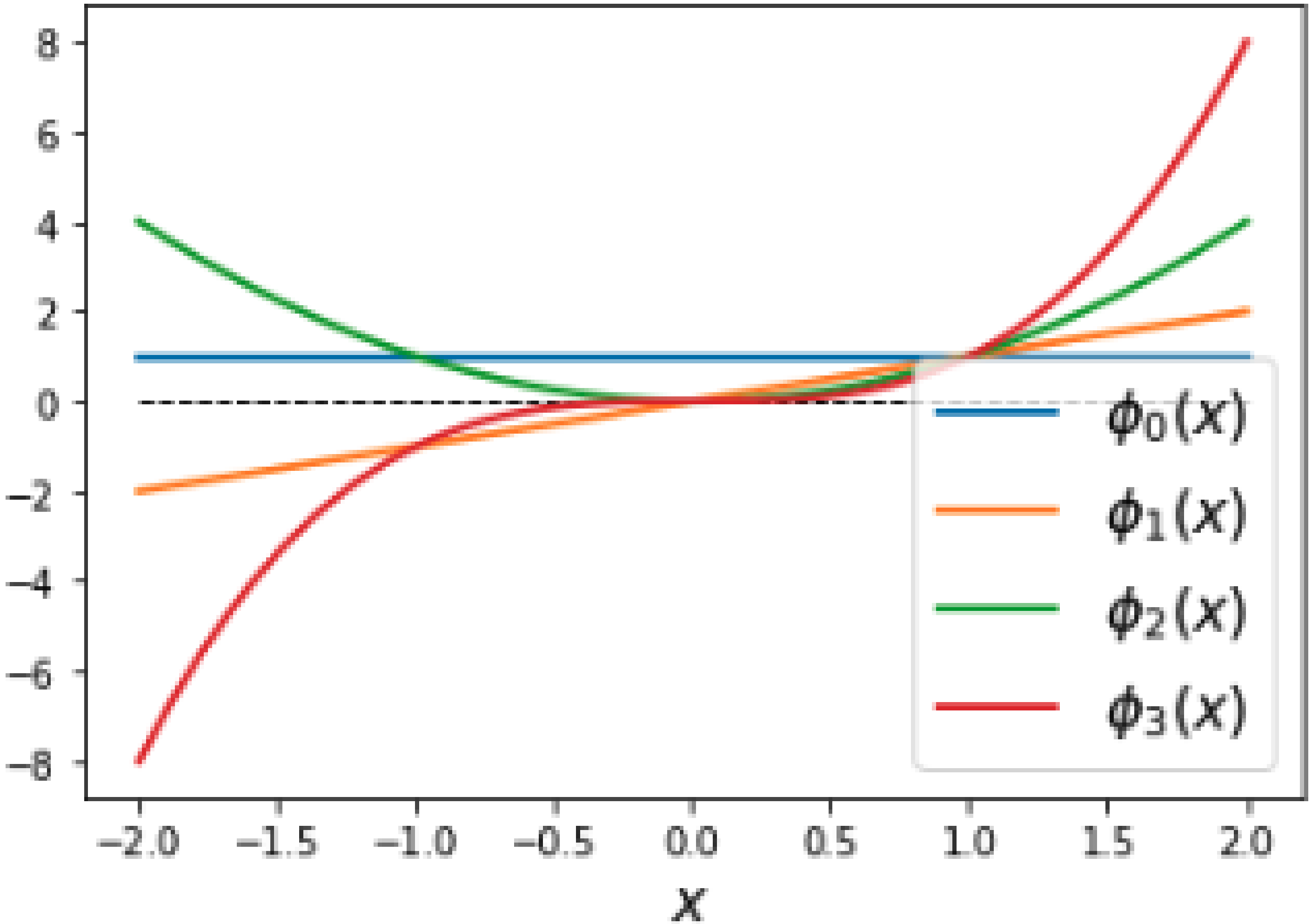


f(x)= $w_0 x + b$

Slope

Intercept

Problem of Linear Regression



NonLinear Regression

- So far, we have been using a linear function for regression:

$$f(x) = w^T x + w_0 = \sum_{i=0}^{d} w_i x_i \quad \text{(Assuming } x_0 = 1\text{)}$$

- Lets generalize this model:

$$f(x) = \sum_{i=0}^{M} w_i \phi_i(x) = w^T \phi(x)$$

where ϕ_i are fixed "basis" functions.

- For linear regression $M = d$, $\phi_i(x) = x_i$.

NonLinear Regression

$$f(x) = \sum_{i=0}^{M} w_i \phi_i(x) = w^T \phi(x)$$

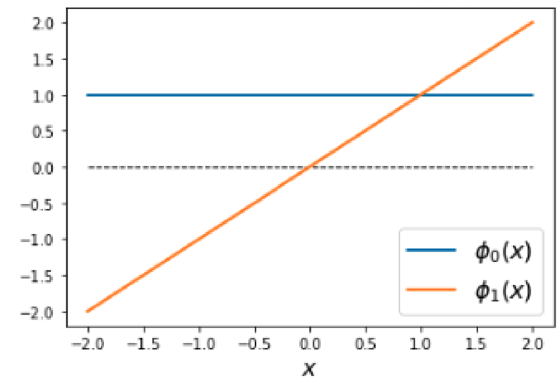
E.g., Polynomial Regression:

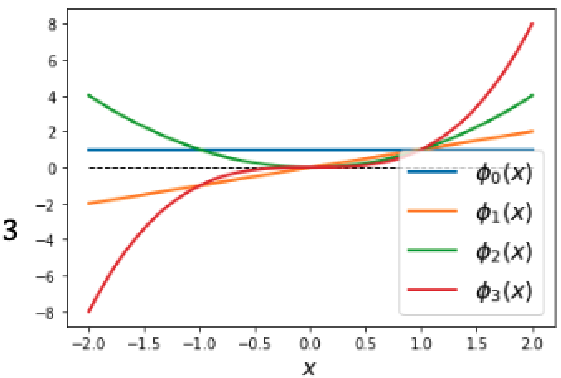
- 1D Polynomial Regression, $\phi(x) = [1, x, x^2, x^3]$:

$$\underset{w}{\text{argmin}} \frac{1}{N} \sum_{n=1}^{N} \left(w^T \phi\left(x^{(i)}\right) - y^{(i)}\right)^2$$

To avoid confusion, note that: $\phi\left(x^{(i)}\right) = \left[1, x^{(i)}, (x^{(i)})^2, (x^{(i)})^3\right]$

$$\mathrm{f}\left(x^{(i)}\right) = w_0 + w_1 x^{(i)} + w_2\left(x^{(i)}\right)^2 + w_3\left(x^{(i)}\right)^3$$





NonLinear Regression

Loss:

$$\underset{w}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} \left(w^T \phi(x^{(i)}) - y^{(i)}\right)^2 = \underset{w}{\operatorname{argmin}} \|\Phi w - y\|^2$$

Where $\Phi = \left[\phi(x^{(1)}), \ldots, \phi(x^{(N)})\right]^T \in \mathbb{R}^{N \times M}$ and $w \in \mathbb{R}^M$.

Optimization:

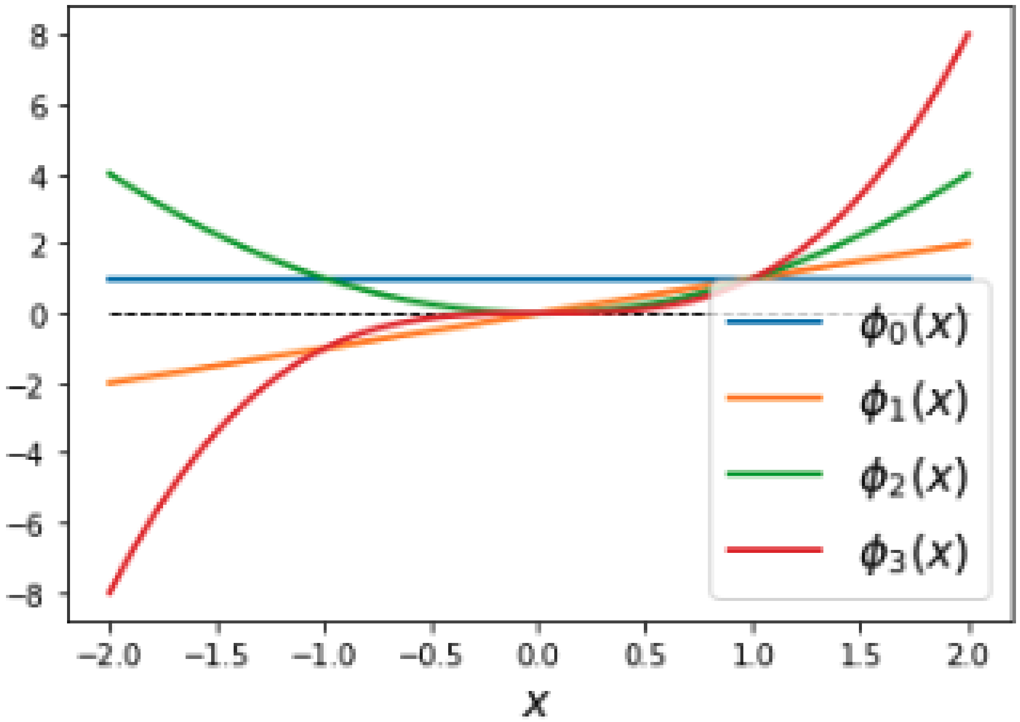
1. Closed form solution: $w^* = (\Phi^T \Phi)^{-1} \Phi^T y$
2. Gradient descent: $w^{(t)} = w^{(t-1)} - \epsilon \nabla_w Loss(w^{(t-1)})$

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

$$W^* = (X^T X)^{-1} X^T Y$$

What is the problem of Nonlinear Regression?

NonLinear Regression



The basis function is all fixed!

Can we learn the basis function?

NonLinear Regression

- Lets first look at what the learning problem might look like:

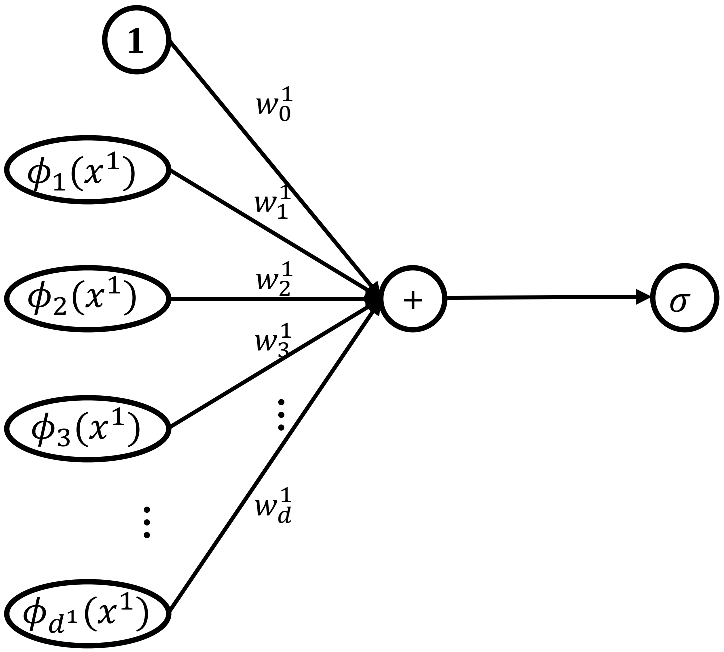
$$\underset{w}{\operatorname{argmin}} \sum_i \left(\left(\sum_j w_j \phi_j \left(x^{(i)} \right) \right) - y^{(i)} \right)^2$$

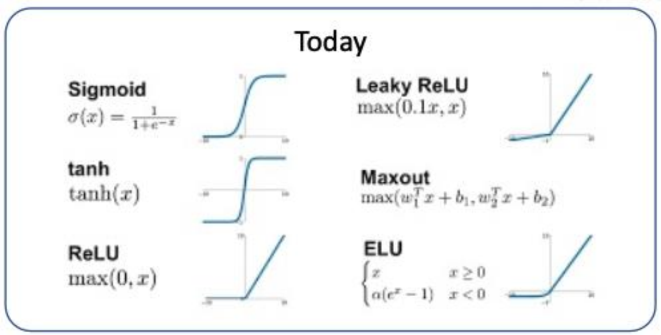
Neural Networks do this for us!

What things are learned here?
What things are fixed here?

NonLinear Regression

1-layer Multi-layer Perceptron





$$\sum_{j=0}^{d^1} w_j^1 \phi_j(x^1)$$

NonLinear Regression

- Lets first look at what the learning problem might look like:

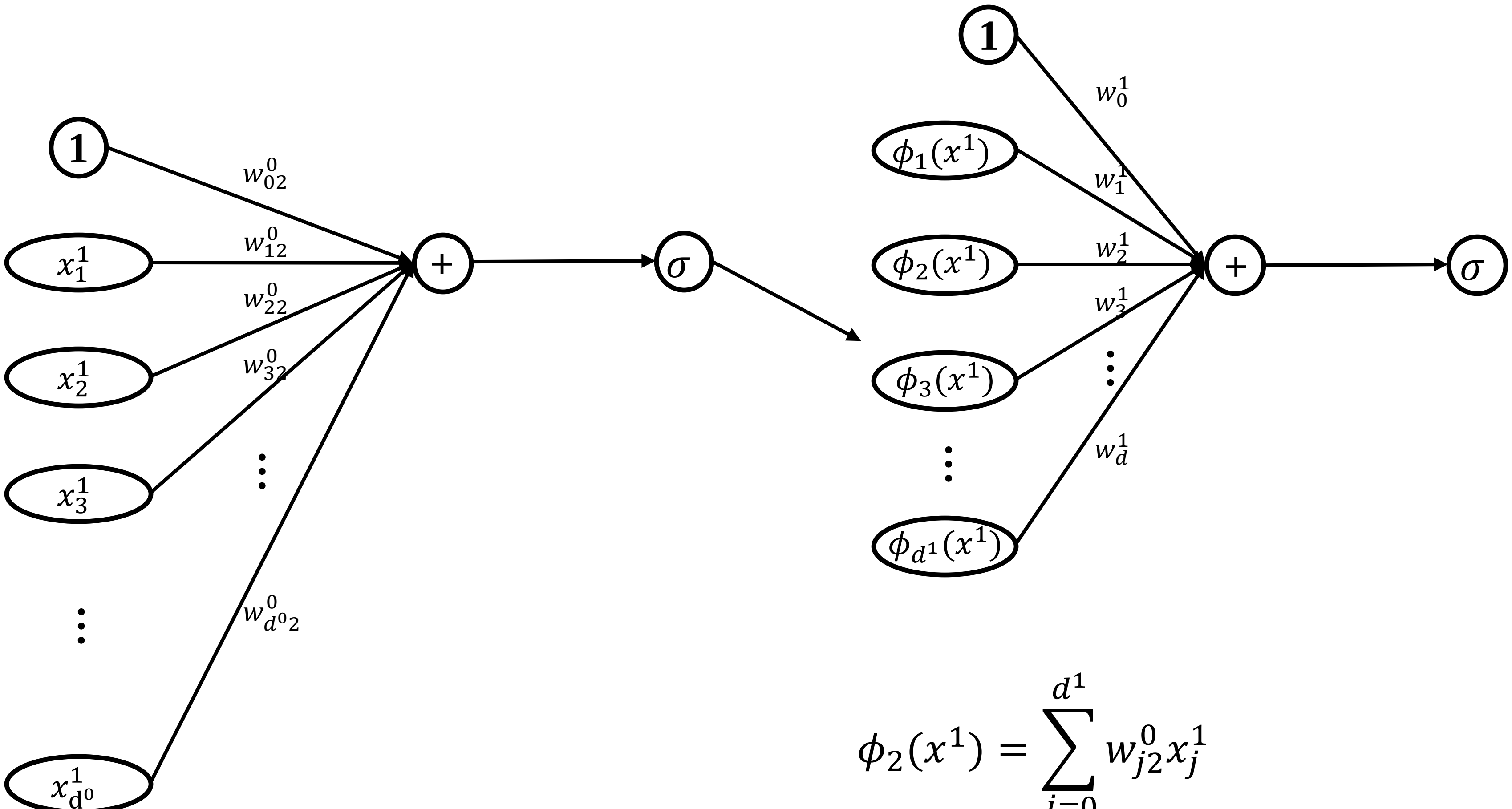
$$\underset{w, \boxed{\{\phi_j\}_{j=1}^{M}}}{\operatorname{argmin}} \sum_i \left(\left(\sum_j w_j \phi_j\left(x^{(i)}\right) \right) - y^{(i)} \right)^2$$

Neural Networks do this for us!

What things are learned here?
What things are fixed here?

NonLinear Regression

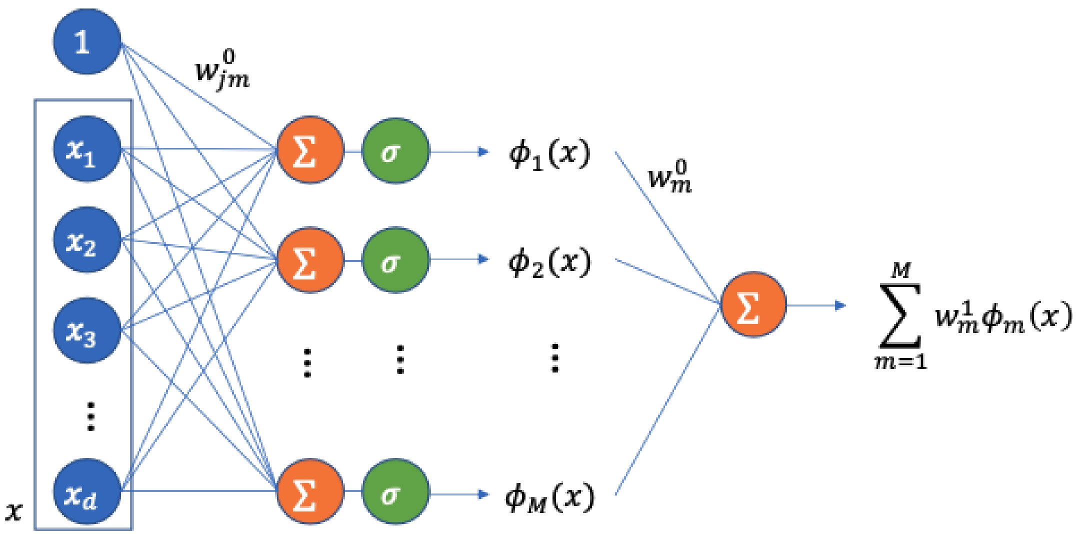
1-hidden layer Multi-layer Perceptron



$$\phi_2(x^1) = \sum_{j=0}^{d^1} w_{j2}^0 x_j^1$$

NonLinear Regression

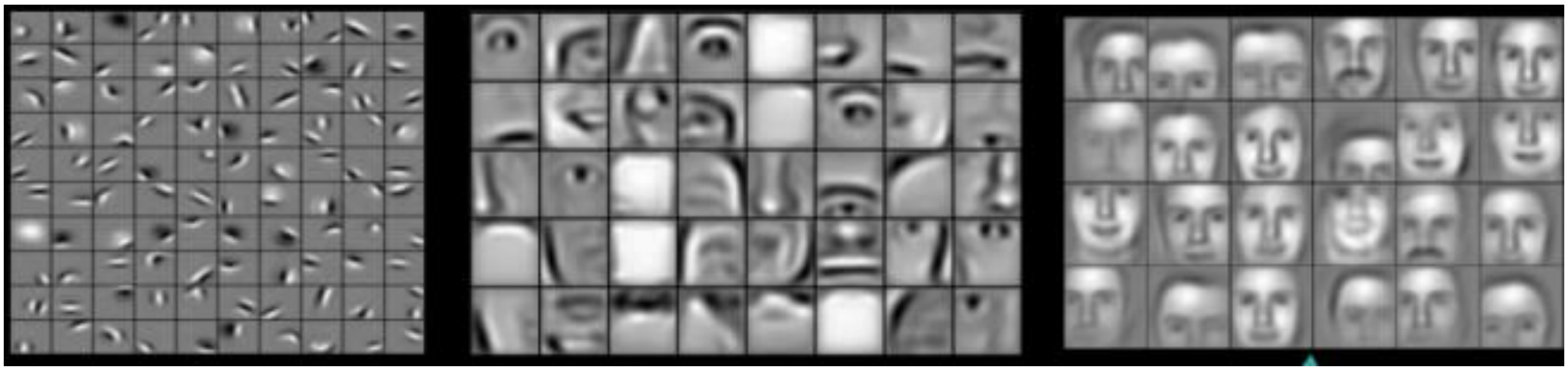
1-hidden layer Multi-layer Perceptron



$$f(x) = \sum_{m=1}^{M} w_m^1 \sigma\left(\sum_{j=1}^{d} W_{jm}^0 x_j\right)$$

Example

- **Activations at hidden layers can be viewed as features extracted as functions of inputs**
- **Every hidden layer represents a level of abstraction**
  - ***Complex features are compositions of simpler features***



- **Number of layers is known as depth of ANN**
  - ***Deeper networks express complex hierarchy of features***

Question?

1. "Judge a man by his questions rather than by his answers."
   – Voltaire
2. "If I had an hour to solve a problem, I'd spend 55 minutes thinking about the problem and 5 minutes thinking about solutions."
   – Albert Einstein
3. "The art and science of asking questions is the source of all knowledge."
   – Thomas Berger
4. "Asking the right questions takes as much skill as giving the right answers."
   – Robert Half
5. "The wise man doesn't give the right answers, he poses the right questions."
   – Claude Lévi-Strauss
6. "Great questions make great companies."
   – Peter Drucker