NEURAL NETWORK

- Main idea:
  - Recall the framework we use to find optimal non-linear function:

$$f(x) = \sum_{\ell=1}^{m} \alpha_\ell \, \phi_\ell(x)$$

where $\begin{cases} m: \text{ number of basis functions (you decide)} \\ \alpha_\ell: \text{ coefficient of each basis function} \\ \phi_\ell: \text{ basis function} \end{cases}$

  - And in the last unit, we know kernel method can be used as basis function:

$$f(x) = \sum_{i=1}^{N} \alpha_i \, k(x, x_i)$$

where $\begin{cases} N: \text{ number of data points} \\ \alpha_i: \text{ coefficient of each basis function} \\ k(\cdot,\cdot): \text{ kernel method (similarity function)} \end{cases}$

  - But, we want our basis function to be adaptive to capture more patterns, that is where neural networks comes in:

Neural Networks

- Give basis functions their own weights and wrap them around an activation function:

$$\phi(x) = \sigma\left(\sum_{i=1}^{d} \omega_i x_i + \beta\right)$$

$$= \sigma\left(\begin{bmatrix} \omega_1 \\ \vdots \\ \omega_d \end{bmatrix}^T \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} + \beta \right)$$

$$= \sigma\left(\begin{bmatrix} w \\ \beta \end{bmatrix}^T \begin{bmatrix} x \\ 1 \end{bmatrix} \right)$$

$$\approx \sigma(w^T x)$$

- And decide the number of neurons you want (lets say m neurons), then the overall function becomes:

$$f(x) = \sum_{\ell=1}^{m} \alpha_\ell \, \sigma(w_\ell^T x)$$

$$= \underbrace{a^T}_{(1\times m)} \sigma(\underbrace{W^T}_{(m\times d)} \underbrace{x}_{(d\times 1)})$$

<matrix form>

Solution to Neural Networks

- We need to optimize 2 parameters: $\begin{cases} \text{coefficients vector } a \\ \text{weights matrix } W \end{cases}$
- By minimizing mean square loss function:

$$a, W = \underset{a, W}{\operatorname{argmin}} \; E_D\left[\left(y - a^T \sigma(W^T x)\right)^2 \right]$$

$$\overset{\text{explicit}}{=} \underset{a, W}{\operatorname{argmin}} \; \frac{1}{N} \sum_{i=1}^{N} \left[\left(y_i - \sum_{l=1}^{m} \alpha_l \, \sigma(w_l^T x) \right)^2 \right]$$

- There is no closed form solution (explain later). So we use numerical method like gradient descent or stochastic gradient descent.
  Recall in gradient descent, we have a learning rate η.
  In each iteration, we update a and W:

$$\begin{cases} a^{t+1} = a^t - \eta \nabla_a E_D\left[(y - a^T \sigma(W^T x))^2 \right] \\ W^{t+1} = W^t - \eta \nabla_W E_D\left[(y - a^T \sigma(W^T x))^2 \right] \end{cases}$$

Why there is almost no closed form solution?

- The loss function we try to optimize is almost alway non-convex.
  Consider a simple neural networks with 2 neurons with the same coefficient $(\alpha_1 = \alpha_2 = 1)$, and different weight vectors $(w_1 \neq w_2)$

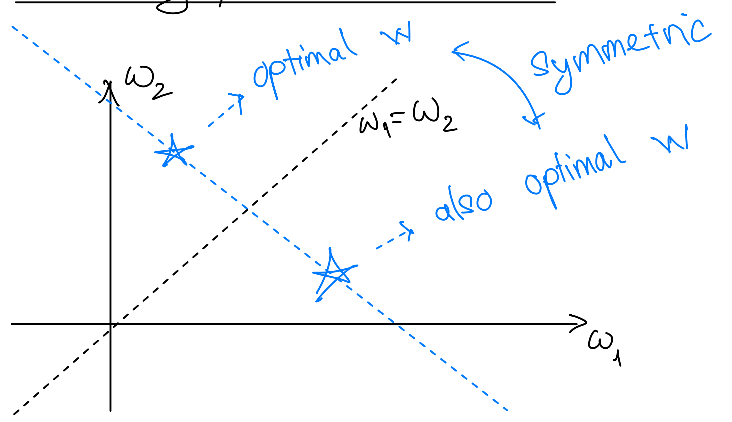
  if $w_1 = w_2$ then they are the same neuron

$$f(x) = \sigma(w_1^T x) + \sigma(w_2^T x)$$

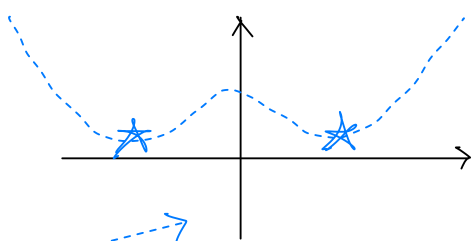
is equal to: $f'(x) = \sigma(w_2^T x) + \sigma(w_1^T x)$ $\quad W = \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix}$

And so the optimal weight matrix $W^* = [w_1^*, w_2^*]$ has 2 symmetrical solutions

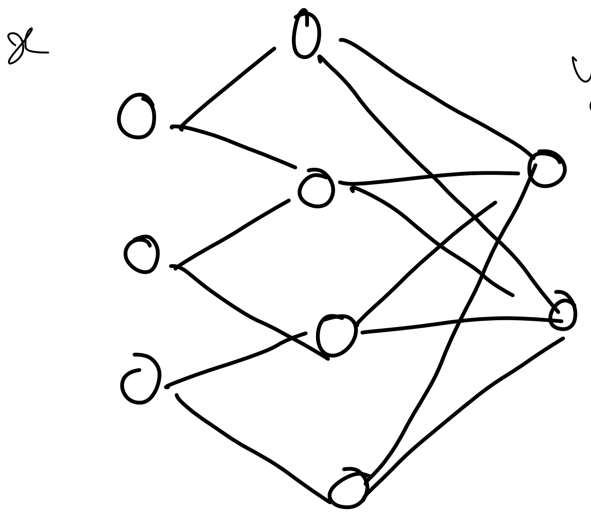
Visually, this means:



(slicing this, we'd get)



- Conclusion: $f(x)$ is $\begin{cases} \text{non-convex} & \text{if \# neurons} > 1 \\ \text{convex} & \text{otherwise} \end{cases}$



basis func

$$y = f(x) = \sum_{\ell}^{N} \theta_\ell \, \phi_\ell(x)$$

And the basis func is fixed.

We want adaptive func.

Hence, give it some of its own weights

$$\phi(x) = \phi\left(\begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}\right) = w_1 x_1 + w_2 x_2 + \dots + w_d x_d$$

$$= W^T x$$

And wrap it around some activation func

$$\phi_\ell(x) = \sigma(w_\ell^T x)$$

So the function becomes:

$$f(x) = \sum_{\ell}^{N} \theta_\ell \, \sigma(w_\ell^T x)$$

$(d \times 1)$

number of neurons

$$= \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_N \end{bmatrix}^T \sigma\left(\begin{bmatrix} w_1^T \\ \vdots \\ w_N^T \end{bmatrix} x\right)$$

one neuron

$$= \theta^T \sigma(W^T x)$$

$1 \times N$ $\quad N \times d$ $\quad d \times 1$

Minimize loss func to get the best $\hat{\theta}$ and $\hat{W}$

$$\hat{\theta}, \hat{W} = \underset{\theta, W}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \left[y_i - \theta^T \sigma(W^T x_i) \right]$$

↳ Solve by gradient descent or stochastic gradient descent